Graph Colourings and Games



Kitty Meeks

Merton College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Michaelmas Term, 2012

Acknowledgements

First and foremost, my thanks go to my supervisor, Prof. Alex Scott: nobody could have been a more helpful and encouraging supervisor, and I am deeply grateful for all his support throughout my time in Oxford. I am also indebted to Prof. Colin McDiarmid and Prof. Dominic Welsh for their help during the processes of transfer and confirmation of status, and to all the other members of the Combinatorics group, both faculty and students, for helping to make my time here so enjoyable.

Thanks are also due to the Engineering and Physical Sciences Research Council, whose financial support made this thesis possible. Last, but certainly not least, I am indebted to my family and to Matt for their encouragement and support over the last three years.

Statement of Authorship

Chapters 2, 4, 5, 6 and 7 are based on joint work with Prof. Alex Scott. Otherwise, to the best of my knowledge, no material in this thesis has previously been written or published by any other person or persons, except where duly acknowledged in the text.

No part of this thesis has been accepted or is currently being submitted for any degree, diploma or certificate or other qualification in this University or elsewhere.

Abstract

Graph colourings and combinatorial games are two very widely studied topics in discrete mathematics. This thesis addresses the computational complexity of a range of problems falling within one or both of these subjects.

Much of the thesis is concerned with the computational complexity of problems related to the combinatorial game (Free-)Flood-It, in which players aim to make a coloured graph monochromatic ("flood" the graph) with the minimum possible number of flooding operations; such problems are known to be computationally hard in many cases. We begin by proving some general structural results about the behaviour of the game, including a powerful characterisation of the number of moves required to flood a graph in terms of the number of moves required to flood its spanning trees; these structural results are then applied to prove tractability results about a number of flood-filling problems. We also consider the computational complexity of flood-filling problems when the game is played on a rectangular grid of fixed height (focussing in particular on $3 \times n$ and $2 \times n$ grids), answering an open question of Clifford, Jalsenius, Montanaro and Sach.

The final chapter concerns the parameterised complexity of list problems on graphs of bounded treewidth. We prove structural results determining the list edge chromatic number and list total chromatic number of graphs with bounded treewidth and large maximum degree, which are special cases of the List (Edge) Colouring Conjecture and Total Colouring Conjecture respectively. Using these results, we show that the problem of determining either of these quantities is fixed parameter tractable, parameterised by the treewidth of the input graph. Finally, we analyse a list version of the Hamilton Path problem, and prove it to be W[1]-hard when parameterised by the pathwidth of the input graph. These results answer two open questions of Fellows, Fomin, Lokshtanov, Rosamond, Saurabh, Szeider and Thomassen.

Contents

1 Introduction **1**

1.1 Flood-filling Games 2

1.2 Edge Colourings and Total Colourings 6

1.3 Parameterised Complexity 9

1.3.1 Treewidth and Pathwidth 11

1.4 Structure of the Thesis 13

2 Paths and Flood-filling Games **15**

2.1 Notation and Definitions 15

2.2 Basic results about paths 17

2.3 Connecting pairs of vertices 19

3 Spanning Trees and Flood-filling Games **26**

3.1 The Main Theorem 27

3.2 Corollaries 38

4 Efficient Algorithms for Flood-filling Problems **45**

4.1 The Complexity of 2-FREE-FLOOD-IT on arbitrary graphs 46

4.2 Graphs with Polynomial Bounds on the Number of Connected Subgraphs 48

4.2.1 The Free-Flood-It case 50

4.2.2 The Fixed-Flood-It case 53

4.3 Connecting Subsets of Vertices 54

4.3.1 The complexity of 2-Linking-Flood-It 54

4.3.2 The complexity of k-Linking-Flood-It 59

5 Flood-It Played on Rectangular Boards of Fixed Height **69**

5.1 Hardness of Fixed-Flood-It and Free-Flood-It on $3 \times n$ boards 72

5.1.1 The 4-Fixed-Flood-It case 72

5.1.2 The 4-Free-Flood-It case 75

5.2 Hardness of Free-Flood-It on $2 \times n$ boards 77
5.2.1 Auxiliary results 78
5.2.2 The reduction 84

6 The parameterised complexity of c-Free-Flood-It on $2 \times n$ boards 90
6.1 Background and auxiliary results 92
6.2 The algorithm 110

7 The Parameterised Complexity of List Problems on Graphs of Bounded Treewidth 121
7.1 List Colouring Problems 124
7.2 List Hamilton Path 133

8 Conclusions 142

Bibliography 145

Chapter 1

Introduction

Graph colourings and combinatorial games are two widely studied topics in discrete mathematics (see for example [9, 26, 35, 47]), with numerous applications. This thesis addresses the computational complexity of a range of problems falling within one or both of these subjects. When considering the complexity of these problems, our first goal is typically to determine whether the problem can be solved efficiently with an algorithm that runs in time bounded by a polynomial function of the input, or whether the problem is NP-complete, in which case it is unlikely that there exists such a polynomial-time algorithm. For some of the problems that are known to be NP-complete, we further investigate their parameterised complexity, as discussed below in Section 1.3. Further background on computational complexity and NP-completeness can be found in [7, 30].

A graph $G = (V, E)$ is a pair of sets, where V is the *vertex set* and $E \subseteq V^{(2)}$ is the *edge set* of G. We write $|G|$ for the number of vertices in G. Other basic graph notation and terminology will follow [13]. Throughout the thesis, all graphs are assumed to be simple graphs, i.e. without loops or multiple edges. A colouring of a graph can be a colouring of the vertices or edges of G, or of both. Most commonly, a colouring of the graph G refers to a *vertex-colouring*, that is a function $\omega : V \to C$,

where C is some set of colours. We say that ω is a *proper* colouring of the vertices of G if, for all $u, v \in V$ such that $uv \in E$, $\omega(u) \neq \omega(v)$. (Edge colourings and total colourings, in which the edges of the graph are coloured instead of or in addition to the vertices respectively, are discussed in Section 1.2 below.)

Much of this thesis is concerned with problems related to flood-filling games, a family of games involving graph colourings, which have been a popular subject for theoretical research in recent years. The best known flood-filling game is the one-player combinatorial game Flood-It, in which the game is played on a square grid: implementations of this game are widely available online, and include a flash game [3] as well as popular smartphone apps [1, 2]. Other versions include Mad Virus [4], played on a hexagonal grid, and the Honey Bee Game [15], a two-player version which is again played on a hexagonal grid. All these games can more generally be regarded as being played on a vertex-coloured graph, and we consider this generalisation of the games in our analysis below.

1.1 Flood-filling Games

Flood-filling games were first introduced as a subject for theoretical research by Arthur, Clifford, Jalsenius, Montanaro and Sach in [8] and Fleischer and Woeginger in [24]. Arthur et. al. studied the complexity of the game Flood-It, which is played on a board consisting of an $n \times n$ grid of coloured squares, where each square is given a colour from some fixed colour-set. The player makes a move by changing the colour of the monochromatic path-connected area containing the top left square, and the goal is to "flood" the board, or make the entire board monochromatic, with the minimum possible number of such moves. Figure 1.1 illustrates a sequence of moves to flood a 4×4 board.

We can more generally regard all flood-filling games as being played on vertex-

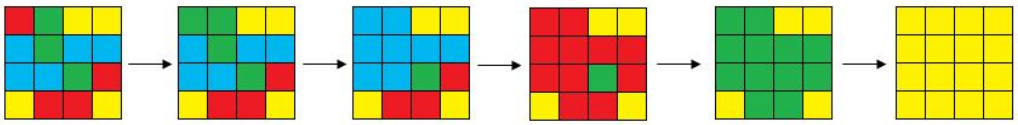


Figure 1.1: The game Flood-It, played on a 4×4 board.

coloured connected graphs. A move then consists of picking a vertex v and a colour d, and giving all vertices in the same monochromatic component as v colour d (where u and v are in the same monochromatic component of the graph G if and only if there is a path from u to v in G in which all vertices, including u and v, have the same colour). The goal is then to make the entire graph monochromatic with as few such moves as possible.

Flood-It, Mad Virus and the Honey Bee Game (studied by Fleischer and Woeginger [24]) are all examples of "fixed" flooding games, in which the player must make every move at the same fixed vertex, so at each move the only choice is which colour to play. Much of our analysis below is concerned instead with the "free" version of such games in which, at each move, the player can also choose freely the vertex at which to play. When this free variant is played on a planar graph, it can be regarded as modelling repeated use of the flood-fill tool in Microsoft Paint.

For any coloured graph, we define the following problems.

- Free-Flood-It is the problem of determining the minimum number of moves required to flood the graph, if we are allowed to make moves anywhere in the graph.

- Fixed-Flood-It is the same problem when all moves must be played at a single specified vertex.[1]

- c-Free-Flood-It and c-Fixed-Flood-It are the variants of Free-Flood-It and Fixed-Flood-It respectively in which only colours from some fixed set

[1] Fixed-Flood-It is often referred to as simply Flood-It, but we use the longer name to avoid confusion with the free version.

of size c are used.

Some key complexity results for these problems are summarised in Table 1.1. Note that we can trivially flood a graph of order n with $n-1$ moves, even in the fixed version; on the other hand, if c colours are present in the initial colouring then in either version we will require at least $c-1$ moves. This lower bound of $c-1$ is always attained if the underlying graph is a clique. Complete bipartite graphs can always be flooded in at most c moves; such graphs can in fact be flooded in $c-1$ moves either if there is a colour that appears at only one vertex, or if there is a colour that appears in both sets of the bipartition, and in all other cases will require exactly c moves.

A related problem, which naturally arises when considering algorithms for Flood-It, is to consider the number of moves required to connect a given subset of the vertices.

- k-Linking Flood-It is the problem, given a subset U of at most k vertices, of determining the minimum number of moves required to create a monochromatic component containing U. The number of colours may be unbounded.

These problems are known to be computationally difficult in many situations. In [8], Arthur, Clifford, Jalsenius, Montanaro and Sach proved that c-Free-Flood-It is NP-hard in the case of a square $n \times n$ board, for every $c \geq 3$, and that this result also holds for the fixed variant. Lagoutte, Noual and Thierry [39, 40] showed that the same result holds when the game is played instead on a hexagonal grid, as in Mad Virus or a one-player version of the Honey Bee Game. Fleischer and Woeginger [24] proved that c-Fixed-Flood-It remains NP-hard when restricted to trees, for every $c \geq 4$,[2] and Fukui, Nakanishi, Uehara, Uno and Uno [27] demonstrated that this result can be extended to show the hardness of c-Free-Flood-It in the same setting.

[2]Note that this proof does in fact require four colours, not three as stated in a previous version of [24].

	Fixed-Flood-It	c-Fixed-Flood-It	Free-Flood-It	c-Free-Flood-It
$n \times n$ boards	NP-h [8]	NP-h for $c \geq 3$ [8]	NP-h [8]	NP-h for $c \geq 3$ [8]
Hexagonal grids	NP-h [39, 40]	NP-h for $c \geq 3$ [39, 40]	NP-h [39, 40]	NP-h for $c \geq 3$ [39, 40]
Trees	NP-h [24]	NP-h for $c \geq 4$ [24]	NP-h [27]	NP-h for $c \geq 4$ [27]
Graphs with polynomially many connected subgraphs	**P [Ch. 4]**	**P [Ch. 4]**	**P [Ch. 4]**	**P [Ch. 4]**
$3 \times n$ boards	NP-h [16]	**NP-h for $c \geq 4$ [Ch. 5]**	NP-h [16]	**NP-h for $c \geq 4$ [Ch. 5]**
$2 \times n$ boards	P [16]	P [16]	**NP-h [Ch. 5]**	**P for c fixed [Ch. 6]**

Table 1.1: The complexity status of flood-filling problems on some key classes of graphs (NP-h = NP-hard, P = Polynomial-time).

In this thesis, we prove a number of positive results concerning the complexity of the games. We show that 2-Free-Flood-It can be solved in polynomial time on arbitrary graphs, answering an open question from [8], and that Free-Flood-It is polynomially solvable on paths, even if the colour-set has unbounded size. We further show that both Fixed-Flood-It and Free-Flood-It can be solved in polynomial time on graphs which have only a polynomial number of connected subgraphs, and that k-Linking-Flood-It can also be solved in polynomial time on arbitrary graphs for any fixed k.

A major focus of research related to these games has been the complexity of the problems when restricted to rectangular boards of fixed height. We show that, although an additive approximation for c-Free-Flood-It on $k \times n$ boards can be computed in polynomial time for any fixed k and c, solving either c-Free-Flood-It or c-Fixed-Flood-It exactly remains NP-hard on $k \times n$ boards, whenever $k \geq 3$ and $c \geq 4$, improving on the result of Clifford et. al. [16] that Free-Flood-It is NP-hard on $3 \times n$ boards. We also investigate the complexity of the problems played

on $2 \times n$ boards, showing that FREE-FLOOD-IT is NP-hard in this setting, but that c-FREE-FLOOD-IT is fixed parameter tractable with parameter c; these results answer an open question of Clifford et. al [16], who showed that FIXED-FLOOD-IT can be solved in linear time on $2 \times n$ boards.

A crucial theorem used in the proof of many of these results is a characterisation of the number of moves required to flood a given graph G: this is equal to the minimum, taken over all spanning trees T of G, of the number of moves required to flood T. This characterisation gives a straightforward framework in which to prove a number of useful facts about the behaviour of flood-filling games.

1.2 Edge Colourings and Total Colourings

As mentioned above, a colouring of a graph can be a colouring of the vertices (a *vertex colouring*), of the edges (an *edge colouring*), or of both the vertices and the edges (a *total colouring*). Proper vertex colourings were defined above, but we can also define proper edge colourings and proper total colourings: a proper edge-colouring of a graph G is an assignment of colours to the edges of G so that no two incident edges receive the same colour, whereas a proper total colouring of a graph $G = (V, E)$ is an assignment of colours to the vertices and edges of G such that no two adjacent vertices or incident edges have the same colour, and no edge has the same colour as either of its endpoints.

For any graph G the *chromatic number*, $\chi(G)$, is defined to be the smallest integer c such that there exists a proper vertex colouring $\omega : V(G) \to \{1, \ldots, c\}$. Similarly, we define the *edge chromatic number*, $\chi'(G)$, to be the smallest integer c such that there exists a proper edge colouring $\omega : E(G) \to \{1, \ldots, c\}$, and the *total chromatic number*, $\chi_T(G)$ is the smallest integer c such that there exists a proper total colouring $\omega : V(G) \cup E(G) \to \{1, \ldots, c\}$.

List colouring is a widely studied generalisation of the standard types of colouring described above. When considering list vertex colouring we are given, as well as a graph G, a collection $\mathcal{L} = (L_v)_{v \in V(G)}$ of lists of permitted colours for each vertex, and a *proper list colouring* of $(G, \mathcal{L})$ is a proper vertex colouring of G in which each vertex v receives a colour from its list L_v. We define *proper list edge colourings* and *proper list total colourings* analogously, when we are given a collection of lists for each edge, or for each vertex and edge respectively.

There are also chromatic numbers associated with list colouring. The *list chromatic number* $\mathrm{ch}(G)$ of a graph G (sometimes called the *choice number* of G) is the smallest integer c such that, whenever $\mathcal{L} = (L_v)_{v \in V(G)}$ is a collection of lists with $|L_v| \geq c$ for every v, there exists a proper list colouring of $(G, \mathcal{L})$; it may be that $\mathrm{ch}(G)$ is strictly larger than $\chi(G)$, as illustrated in Figure 1.2. Once again, for any graph G we define the *list edge chromatic number*, $\mathrm{ch}'(G)$, and *list total chromatic number*, $\mathrm{ch}_T(G)$, in the same way.

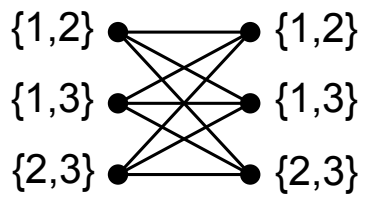


Figure 1.2: The list chromatic number may be larger than the chromatic number: this graph has chromatic number two, but there is no proper list colouring using the illustrated lists of length two.

Of course, there is a correspondence between edge and total colouring problems and vertex colouring problems. For any graph $G = (V, E)$, the *line graph* $L(G)$ of G has vertex set E, and $e, f \in E$ are adjacent in $L(G)$ if and only if e and f are incident in G. Then a proper vertex colouring of $L(G)$ corresponds exactly to a proper edge colouring of G. Similarly, we define the *total graph* $T(G)$ of G to have vertex set $V \cup E$, and edge set $E \cup \{ef : e, f \in E \text{ and } e, f \text{ incident in } G\} \cup \{ve : v \text{ is an endpoint of } e\}$, and proper vertex colourings of $T(G)$ correspond exactly to proper total colourings

of G.

It is straightforward to obtain some upper and lower bounds on the (list) edge and total chromatic numbers of a graph G. If we let $\Delta(G)$ denote the maximum degree of G, then the following results are 'folklore':

$$\Delta(G) \leq \chi'(G) \leq \mathrm{ch}'(G) \leq 2\Delta(G) - 1,$$

and

$$\Delta(G) + 1 \leq \chi_T(G) \leq \mathrm{ch}_T(G) \leq 2\Delta(G) + 1.$$

In both cases the lower bound can be seen by considering the number of colours required to properly colour all edges incident with a single vertex (and the vertex itself in the case of total colouring). It is clear that the list edge or list total chromatic number is at least the edge or total chromatic number, if we consider the instance in which all colour-lists are identical. The upper bound is the minimum length of lists required to ensure that we can always colour the graph greedily, i.e. by choosing the first colour in the list for each object that is not already used on an adjacent or incident vertex or edge.

For the edge chromatic number, we have the much stronger result of Vizing:

Theorem 1.1 ([50])**.** *Let G be a simple graph. Then $\chi'(G)$ is equal to either $\Delta(G)$ or $\Delta(G) + 1$.*

There is no direct analogue of Vizing's theorem for the list edge chromatic number. The *List (Edge) Colouring Conjecture*(discussed in [5, 12, 31]) asserts that $\chi'(G) = \mathrm{ch}'(G)$ for any graph G, and would immediately imply Vizing's conjecture (1976) that $\mathrm{ch}'(G) \leq \Delta(G) + 1$. However, neither of these conjectures has been proved except for certain special classes of graphs, and the best general bound on the list edge chromatic number is due to Kahn.

Theorem 1.2 ([36]). *For any $\epsilon > 0$, if $\Delta(G)$ is sufficiently large,*

$$\mathrm{ch}'(G) \leq (1+\epsilon)\Delta(G).$$

In Chapter 7, we prove a special case of the List Colouring Conjecture, and apply this to the computational complexity of determining the list edge chromatic number.

Another long-standing but unproved conjecture concerns the list total chromatic number: the *Total Colouring Conjecture* [10, 51] asserts that

$$\chi_T(G) \leq \Delta(G) + 2.$$

A weakened form of this conjecture was proved by Molloy and Reed [45]: they showed that $\chi_T(G) \leq \Delta + 10^{26}$, for sufficiently large Δ. In Chapter 7, we prove a special case of the Total Colouring Conjecture, which can be applied to the computational complexity of determining the list total chromatic number.

1.3 Parameterised Complexity

The theory of parameterised complexity provides a framework in which to analyse more precisely the behaviour of problems that are known to be computationally hard in a classical complexity setting. When determining whether a problem belongs to **P**, we consider the running time as a function only of the input size, but in the study of parameterised complexity we also consider how the running time of algorithms depends on one or more *parameters*.

Typically we are interested in situations where the parameters considered can be assumed to be small compared with the overall input size. Thus, algorithms whose running time is exponential only in the small parameter are much more useful than those whose running time is exponential in the input size. A standard example of

such a situation is the complexity of computing a database query, when the size of the query can reasonably be assumed to be much smaller than that of the entire database. Another example arises when considering flood filling games: we may be interested in the complexity of computing the length of an optimal sequence to flood a graph of order n (where n is large) that is coloured with some small fixed number of colours, c, and we can take c to be the parameter.

Problems which can be solved efficiently when a given parameter is small are said to be *fixed parameter tractable*. More precisely, an algorithm for the problem Π is an *FPT-algorithm* for the parameter κ if, for any instance I of Π, the running time is at most $f(\kappa(I)) \cdot p(|I|)$, for some polynomial p and computable function f (where $|I|$ is the total input size, and $\kappa(I)$ is the size of the parameter for this instance). (Π, κ) is then said to be fixed parameter tractable if such an algorithm exists. The class of all fixed parameter tractable problems is denoted by FPT.

A standard method of showing that a problem is unlikely to belong to FPT (and so the best algorithm has running time $|I|^{f(\kappa(I))}$ for some unbounded function f) is to prove that it is W[1]-hard: this is the analogue of proving a problem to be NP-complete and implies, up to similar complexity theoretic assumptions, that the problem is not fixed parameter tractable.

Just as with proving that a problem is NP-hard, the primary method for showing the intractability of a parameterised problem is to give a reduction from another problem that is known to be W[1]-hard. If we know that the parameterised decision problem Π with parameter κ is difficult and want to show that the problem Π' with parameter κ' is just as hard then, given any instance I of Π, we want to be able to construct an instance I' of Π' such that I' is a yes-instance for Π' if and only if I is a yes-instance for Π. The idea is to show in this way that if Π' is fixed parameter tractable (with parameter κ') then Π must also be (with parameter κ). Thus we need to show not only that I' can be computed from I in time polynomial in

$|I|$ and that $|I'|$ is bounded by a polynomial function of $|I|$ (as we would for an NP-hardness reduction), but also that $\kappa'(I')$ is bounded by some function (not necessarily polynomial) of $\kappa(I)$. This kind of parameterised reduction was first introduced by Downey and Fellows [20].

One useful W[1]-hard problem, which is commonly used for such reductions, is Multicolour Clique (shown to be W[1]-hard by Fellows, Hermelin, Rosamond and Vialette in [22]).

Multicolour clique

Input: A k-vertex-coloured graph G.

Question: Does there exist a complete subgraph of G containing exactly one vertex of each colour?

Further background on the theory of parameterised complexity can be found in [21, 25, 46].

1.3.1 Treewidth and Pathwidth

When considering the parameterised complexity of graph problems, two very useful parameters are the treewidth and the pathwidth of the input graph. Given a graph G, we say that $(T, \mathcal{D})$ is a *tree decomposition* of G if T is a tree and $\mathcal{D} = \{\mathcal{D}(t) : t \in T\}$ is a collection of non-empty subsets of $V(G)$ (or *bags*), indexed by the nodes of T, satisfying:

1. $V(G) = \bigcup_{t \in T} \mathcal{D}(t)$,
2. for every $e = uv \in E(G)$, there exists $t \in T$ such that $u, v \in \mathcal{D}(t)$,
3. for every $v \in V$, if $T(v)$ is defined to be the subgraph of T induced by nodes t with $v \in \mathcal{D}(t)$, then $T(v)$ is connected.

The *width* of the tree decomposition $(T, \mathcal{D})$ is defined to be $\max_{t \in T} |\mathcal{D}(t)| - 1$, and the *treewidth* of G is the minimum width over all tree decompositions of G. A *path decomposition* is a tree decomposition $(P, \mathcal{D})$ in which the indexing tree is a path, and the *pathwidth* of G is the minimum width over all path decompositions of G. Note that trees have treewidth one, and paths have pathwidth one.

When we are given a tree decomposition $(T, \mathcal{D})$ of G, we assume that an arbitrary node $r \in V(T)$ is chosen to be the *root* of T, and define the *depth* $\text{depth}(t)$ of any $t \in T$ to be the distance from r to t. For any $v \in V(G)$, we then define t_v to be the unique node t of minimum depth such that $v \in \mathcal{D}(t)$ (i.e. t_v is the node of minimal depth in the subtree $T(v)$).

Graphs having treewidth at most k can alternatively be characterised as *partial k-trees*, as in [6, 53]. A graph is a *k-tree* if and only if it can be obtained from a complete graph on $k+1$ vertices by repeatedly adding a new vertex whose neighbourhood induces a k-clique in the existing graph; a graph is a partial k-tree if it is a subgraph of a k-tree. It follows immediately from this equivalent definition that if a graph G of order n is a partial k-tree (i.e. it has treewidth at most k) then G has at most kn edges.

The following lemma (equivalent to a result from [11]) shows that we may assume that the tree indexing the tree decomposition of a graph of order n has at most n nodes.

Lemma 1.3. *Let G be a graph of order n and treewidth k. Then there exists a width-k tree decomposition $(T, \mathcal{D})$ for G with $|T| \leq n$.*

Proof. Let $(T_0, \mathcal{D}_0)$ be any width k tree decomposition of G. We will transform this decomposition into one with the required properties by repeatedly contracting edges of T_0. For $i \geq 0$, if xy is an edge of T_i such that either $\mathcal{D}_i(x) \subseteq \mathcal{D}_i(y)$ or $\mathcal{D}_i(y) \subseteq \mathcal{D}_i(x)$, we obtain T_{i+1} by contracting the edge xy to a single node z, and set $\mathcal{D}_{i+1}(z) = \mathcal{D}_i(x) \cup \mathcal{D}_i(y)$ (note $|\mathcal{D}_{i+1}(z)| \leq k$), with all other bags of $\mathcal{D}_{i+1}$ the same

as $\mathcal{D}_i$. If there is no such edge xy, we stop and set $(T, \mathcal{D}) = (T_i, \mathcal{D}_i)$.

We claim that the decomposition $(T, \mathcal{D})$, obtained when this process terminates, is as required. Observe that for every node $t \in T$ we must have $t = t_v$ for some $v \in V(G)$: this is trivially true for the root (any vertex in its bag will do), and if any other $t \in T$ is not equal to t_v for some $v \in V$ then all vertices in $\mathcal{D}(t)$ also belong to the bag indexed by the parent of t, and so we would have contracted this edge. Thus we have a surjective mapping from $V(G)$ to $V(T)$, implying that $|T| \leq n$, as required. □

Many graph problems that are known to be NP-hard in general are fixed parameter tractable when parameterised by the treewidth of the graph, and often there even exists a linear-time algorithm to solve the problem on graphs of fixed treewidth [6, 14, 49] (note, however, that this is *not* the case for the flood-filling problems discussed in Section 1.1). Courcelle [18] proved that this is the case for a large family of graph problems, demonstrating that any graph property which is expressible in monadic second order logic can be decided in linear time on graphs of bounded treewidth.

1.4 Structure of the Thesis

The remainder of the thesis is structured as follows. In the next two chapters we prove a number of general results concerning flood-filling games, which will be used in the proofs of our complexity results in later chapters: Chapter 2 focuses on the game played on paths, and on the process of linking a pair of vertices in a given graph, whereas in Chapter 3 we prove and exploit a characterisation of the number of moves required to flood an arbitrary graph in terms of its spanning trees. Chapter 4 is then concerned with problems related to flood-filling games which can be solved in polynomial time, and in Chapters 5 and 6 we consider the complexity of flood-filling games restricted to rectangular boards of fixed height. In Chapter 7, we move away

from the topic of flood-filling games, and instead consider the complexity of some list problems on graphs of bounded treewidth. Finally, in Chapter 8 we summarise the main results of the thesis, and mention some key open problems.

Chapter 2

Paths and Flood-filling Games

In this chapter and in Chapter 3, we prove some general results about flood-filling games, played on arbitrary graphs. In both chapters, we will be considering the "free" variant of the game. As well as being of independent interest, these general results are crucial for the analysis of algorithms presented later in Chapters 4 and 5.

We begin in this chapter by considering flood-filling games played on paths, and prove two useful facts about the behaviour of the games in this setting. We then apply these results to give a characterisation of the number of moves that must be played in an arbitrary graph G in order to link two given vertices u and v: this is equal to the minimum, taken over all u-v paths P in G, of the number of moves required to flood P. Finally, we prove some corollaries which will be useful in later chapters.

Before proving any of these results, we begin in Section 2.1 with some more precise definitions and terminology for discussing flood-filling games.

2.1 Notation and Definitions

Suppose the game is played on a graph $G = (V, E)$, with an initial colouring ω (not necessarily proper) using colours from the *colour-set* C. Each move $m = (v, d)$ then

involves choosing some vertex $v \in V$ and a colour $d \in C$, and assigning colour d to all vertices in the same monochromatic component as v. The goal is to make every vertex in G the same colour, using as few moves as possible.

Let A be any subset of V. We set $m_G(A, \omega, d)$ to be the minimum number of moves, in the free variant, that we must play in G (with initial colouring ω) to create a monochromatic component of colour d that contains every vertex in A, and $m_G(A, \omega) = \min_{d \in C} m_G(A, \omega, d)$. Thus $m_G(G, \omega, d)$ is the minimum number of moves required to give all vertices in G colour d. We say a move $m = (v, d)$ is *played in* A if $v \in A$, and that A is *linked* under ω if it is contained in a single monochromatic component with respect to this colouring. Disjoint subsets $A, B \subseteq V$ are *adjacent* if there exists $ab \in E$ with $a \in A$ and $b \in B$. We denote the set of colours appearing an A under the colouring ω by $\mathrm{col}(A, \omega)$. We will use all the same notation when referring to (the vertex-set of) a subgraph H of G as for a subset $A \subseteq V(G)$.

For any vertex $v \in V$, we write $\mathrm{comp}_G(v, \omega)$ to denote the maximal monochromatic component of G, with respect to ω, that contains v. Given any sequence of moves S on a graph G with initial colouring ω, we denote by $S(\omega, G)$ the new colouring obtained by playing S in G.

Note that, if the initial colouring ω of a graph G is *not* a proper colouring, we can contract each monochromatic component of G to a single vertex, obtaining a coloured graph which is equivalent to G in the following sense: any sequence of moves which floods the original graph will flood this new graph (playing each move at the vertex corresponding to the monochromatic component in which it was originally played), and vice versa. Sometimes it will be convenient to assume that we start with a properly coloured graph. However, there is no guarantee that contracting monochromatic components in this way preserves structural properties of the initial graph, so in the proof of some results later in the thesis we do still need to consider the possibility that the initial colouring is not proper.

2.2 Basic results about paths

We start with a simple but extremely useful monotonicity result for paths, which shows that deleting a section of a path (and joining the remaining sections) cannot increase the number of moves we need to flood the path. If Q is a subpath of the path P, a *path contraction* of Q involves deleting the internal vertices of Q from P and adding a single edge between its endpoints.

Lemma 2.1. *Let P be a path, with colouring ω from colour-set C, and let P' be a second path, obtained from P by performing a path contraction of some subpath of P. Then, for any $d \in C$, $m_{P'}(P', \omega, d) \leq m_P(P, \omega, d)$. We also have $m_{P'}(P', \omega) \leq m_P(P, \omega)$.*

Proof. Fix $d \in C$, and note we may assume that ω is a proper colouring of P (contracting monochromatic components if necessary). We proceed by induction on $m_P(P, \omega, d)$. The result is trivially true for $m_P(P, \omega, d) = 0$, so assume $m_P(P, \omega, d) \geq 1$ and that the result holds for any path Q with colouring ω_Q such that $m_Q(Q, \omega_Q, d) < m_P(P, \omega, d)$. Let S be an optimal sequence to flood P in colour d, and let α be the first move of S. Suppose that P_1 is the segment of P induced by vertices of $V(P) \setminus V(P')$, and that the neighbours of P_1 in $V(P) \setminus V(P_1)$ are w_1 and w_2 (so $V(P') = V(P) \setminus V(P_1)$, and $E(P') = E(P[V(P')] \cup \{w_1 w_2\})$.

First suppose that α is not played at a vertex $u \in V(P_1)$. Then we can play α on P', and the path P' with colouring $\alpha(\omega, P')$ is identical (after possibly contracting monochromatic components) to a path obtained from P with colouring $\alpha(\omega, P)$ by performing a path contraction of some subpath (and again contracting monochromatic components): if α is played at w_1 or w_2 and $\omega(w_1) = \omega(w_2)$ then we remove the segment induced by $V(P_1) \cup \{w_1\}$, and otherwise we simply remove the segment P_1. Moreover, $m_P(P, \alpha(\omega, P), d) < m_P(P, \omega, d)$, and so by the inductive hypothesis we

have $m_{P'}(P', \alpha(\omega, P'), d) \leq m_P(P, \alpha(\omega, P), d)$. Thus

$$m_{P'}(P', \omega, d) \leq 1 + m_{P'}(P', \alpha(\omega, P'), d) \leq 1 + m_P(P, \alpha(\omega, P), d) = m_P(P, \omega, d),$$

as required.

Now suppose α is played at a vertex $u \in V(P_1)$. Then the path obtained from P with colouring $\alpha(\omega, P)$ by deleting P_1 and joining w_1 and w_2 is in fact precisely the path P' with colouring ω, since ω is a proper colouring and so changing the colour of v cannot change the colour of any other vertex. Hence, as $m_P(P, \alpha(\omega, P), d) < m_P(P, \omega, d)$ we have, by the inductive hypothesis,

$$m_{P'}(P', \omega, d) \leq m_P(P, \alpha(\omega, P), d) < m_P(P, \omega, d).$$

Thus in all cases we have $m_{P'}(P', \omega, d) \leq m_P(P, \omega, d)$, and as this holds for any colour $d \in C$ it follows immediately that $m_{P'}(P', \omega) \leq m_P(P, \omega)$. □

We will also need the following simple fact about additivity for paths.

Lemma 2.2. *Let P_1 and P_2 be paths, with colourings ω_1 and ω_2 from colour-set C, let $P = P_1P_2$ be the path obtained by concatenating P_1 and P_2, and let ω be the colouring of P which agrees with ω_i on P_i. Then, for any $d \in C$, $m_P(P, \omega, d) \leq m_{P_1}(P_1, \omega_1, d) + m_{P_2}(P_2, \omega_2, d)$.*

Proof. For $i \in \{1, 2\}$, let S_i be an optimal sequence to make P_i monochromatic with colour d. Suppose we begin by playing the sequence S_1 on P. This makes P_1 monochromatic with colour d, and may also absorb some vertices from P_2. But by Lemma 2.1, we can make P_2', the remainder of P_2, monochromatic in colour d with a sequence T_2 of at most $|S_2|$ moves. In the course of T_2, some vertex on P_2 may absorb P_1, but as this vertex ends up with colour d, the sequence S_1T_2 must ultimately give P_1 colour d. Hence $m_P(P, \omega, d) \leq |S_1| + |S_2| = m_{P_1}(P_1, \omega_1, d) + m_{P_2}(P_2, \omega_2, d)$. □

2.3 Connecting pairs of vertices

In this section we exploit the facts proved in the previous section to analyse sequences of moves to link pairs of vertices in arbitrary graphs. First, however, we need a few further definitions. Given two vertices $u, v \in V$, we define $\mathcal{P}_G(u, v)$ to be the set of all u-v paths in G. If S is a sequence of moves linking two vertices u and v, we say that $P \in \mathcal{P}_G(u, v)$ is *S-critical* with respect to G and ω if, for all x, y lying on P, S does not link x and y in G before they are linked along P (note that, once two vertices are linked in a given graph or subgraph, i.e. they belong to the same monochromatic component of the graph, they will remain linked regardless of what further moves are played). We begin by showing that such a path must always exist.

Lemma 2.3. *Let G be a connected graph, with colouring ω, and suppose that S is a sequence of moves linking the vertices u_1 and w_2. Then there exists an S-critical u_1-w_2 path with respect to G and ω.*

Proof. We proceed by induction on $|S|$. The base case for $|S| = 0$ is trivially true, so we assume $|S| > 0$.

Let m be the first move in S that links u_1 and w_2. Denote by S_1 the initial segment of moves in S occurring before m, and by S_2 those occurring after m, so $S = S_1 m S_2$. Let U be the monochromatic component containing u_1 immediately before m, and W the monochromatic component containing w_2 at this point. There are two cases: either U and W are adjacent, or there is some third monochromatic component V, adjacent to both, such that m changes the colour of V to be the same as that of both U and W.

First suppose we are in the second situation, so m changes the colour of a third component, V, to link U and W. Let S_U, S_V and S_W be the subsequences of S_1 consisting of moves played in U, V and W respectively. Note that these subsequences are disjoint, and that no move from any of the subsequences (when they are interleaved

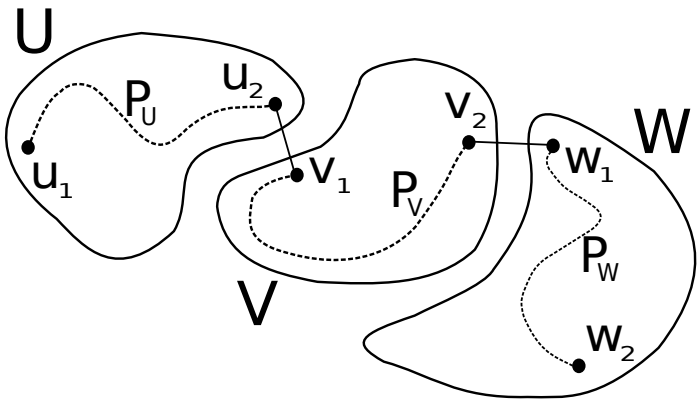


Figure 2.1: The path P

as in S_1) changes the colour of any vertex outside the component (U, V or W) in which it is played. Thus, if we consider S_U played in the isolated graph $G[U]$ (with colouring ω), this has the same effect on the vertices of U (respectively V, W) as if we play S_1 in the larger graph G (and similarly for S_V and S_W played in $G[V]$ and $G[W]$ respectively).

Now pick $u_2 \in U$, $w_1 \in W$ and $v_1, v_2 \in V$ such that u_2 is adjacent to v_1 and v_2 is adjacent to w_1 (note that there must exist at least one possible choice for each of these vertices, as U, V and V, W are adjacent). Clearly if we play S_U in $G[U]$ (with colouring ω) then this links u_1 and u_2, and similarly S_V links v_1 and v_2 in $G[V]$ and S_W links w_1 and w_2 in $G[W]$. Moreover, as each of these sequences is strictly shorter than S, we can apply the inductive hypothesis to obtain a u_1-u_2 path P_U in $G[U]$ such that, when we play S_U, no pair of vertices on P_U is linked in $G[U]$ before it is linked along P_U. In the same way we obtain v_1-v_2 and w_1-w_2 paths P_V, P_U in $G[V]$, $G[W]$ respectively.

Now suppose we play S_1 in G, and consider the effect of these moves on the path $P = u_1 P_U u_2 v_1 P_V v_2 w_1 P_W w_2$, illustrated in Figure 2.1. First observe that P is indeed a path: as U, V and W are disjoint, no vertex may be repeated. We claim that P is the path we require. For, if not, there exist vertices x and y on P and a move $m' \in S$ such that m' links x and y in G before they are linked along P. Clearly m'

cannot be from S_2, as P is monochromatic before any move in S_2 is made, so x and y are already linked along the path by this point. Nor can we have $m' = m$, since if x and y are not already linked along P before m, then m links them along P. So $m' \in S_1$. But then x and y are linked before m, so they must both lie in one of U, V and W (as these three components are not linked immediately before m, and no move can split an existing monochromatic component). Without loss of generality, suppose $x, y \in U$. But then both x and y lie on P_U so, by definition of P_U, they are not linked in $G[U]$ by S_U before they are linked along P_U, and hence (as S_U has the same effect on $G[U]$ as does the sequence S_1 played in G) they are not linked in G before they are linked along P. So P is as required.

Now suppose that in fact U and W are adjacent. We then choose $u_2 \in U$ and $w_1 \in W$ so that u_2 and w_1 are adjacent (again noting that there must exist such a pair of vertices). As before, we apply the inductive hypothesis to obtain suitable paths P_U and P_W in $G[U]$ and $G[W]$ respectively, and by exactly the same reasoning the path $P = u_1 P_U u_2 w_1 P_W w_2$ is as required. □

We now prove the main result of this chapter, that it is enough to consider only u-v paths in G in order to determine $m_G(\{u, v\}, \omega, d)$.

Theorem 2.4. *Let $G = (V, E)$ be any connected graph with colouring ω from colour-set C, and let $u, v \in V$ and $d \in C$. Then*

$$m_G(\{u, v\}, \omega, d) = \min_{P \in \mathcal{P}_G(u,v)} m_P(P, \omega, d).$$

Proof. We will begin by proving that $m_G(\{u, v\}, \omega, d) \geq \min_{P \in \mathcal{P}_G(u,v)} m_P(P, \omega, d)$, and then show that the reverse inequality must also hold.

Let S be an optimal sequence to link u and v with colour d in G, and let P' be a S-critical u-v path with respect to G and ω. Let S' be the subsequence of S consisting

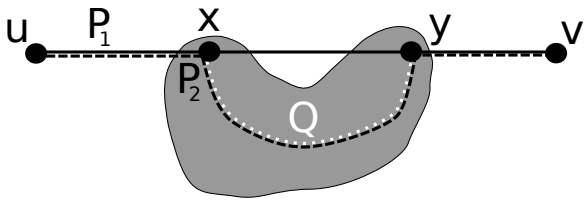


Figure 2.2: The u-v paths P_1 and P_2. The shaded area is a monochromatic component of G with respect to the colouring $\alpha(\omega, G)$.

of moves played in monochromatic components that intersect P', and without loss of generality assume every move in S' is played on P' (otherwise we may replace it with an equivalent move played on P'). Then, as P' is S-critical, the sequence S' played on the path P' (considered as a separate graph) has the same effect on P' as does S when played in G, and so makes P' monochromatic. Thus,

$$m_G(\{u, v\}, \omega, d) = |S| \geq |S'| \geq m_{P'}(P', \omega, d) \geq \min_{P \in \mathcal{P}_G(u,v)} m_P(P, \omega, d).$$

To show the reverse inequality, we prove by induction on $m_P(P, \omega, d)$ that, for any $P \in \mathcal{P}_G(u, v)$, we have $m_G(\{u, v\}, \omega, d) \leq m_P(P, \omega, d)$. The base case, for $m_P(P, \omega, d) = 0$, is trivially true, so let $P_1 \in \mathcal{P}_{uv}$ and suppose S_1 is a nonempty optimal sequence to flood P_1 with colour d. Consider the first move, α, of S_1.

First suppose that there exist two or more vertices on P_1 whose colours are changed by α when the move is played in G, but are not linked along the path. Suppose x is the first such vertex on P_1 when the path is traversed from u to v, and y the last, and observe that there exists a monochromatic x-y path Q in G with colouring $\alpha(\omega, G)$. Let P_2 be the u-v path in G obtained by joining the segments of P_1 from u to x and from y to v with the path Q, as illustrated in Figure 2.2. Then the path P_2 with colouring $\alpha(\omega, G)$ can (after contracting monochromatic components) be obtained from P_1 with colouring $\alpha(\omega, P_1)$ by deleting some consecutive vertices and joining the

resulting segments and so, by Lemma 2.1,

$$m_{P_2}(P_2, \alpha(\omega, G), d) \leq m_{P_1}(P_1, \alpha(\omega, P_1), d) = m_{P_1}(P_1, \omega, d) - 1.$$

Hence, by the inductive hypothesis, we have

$$m_G(\{u, v\}, \alpha(\omega, G), d) \leq m_{P_2}(P_2, \alpha(\omega, G), d),$$

and so

$$\begin{aligned} m_G(\{u, v\}, \omega, d) &\leq 1 + m_G(\{u, v\}, \alpha(\omega, G), d) \\ &\leq 1 + m_{P_2}(P_2, \alpha(\omega, G), d) \\ &\leq 1 + m_{P_1}(P_1, \alpha(\omega, P_1), d) \\ &= m_{P_1}(P_1, \omega, d). \end{aligned}$$

Now suppose that α does not change the colour of any such pair of vertices on P_1. Then $\alpha(\omega, G)|_{P_1} = \alpha(\omega, P_1)|_{P_1}$ and so $m_{P_1}(P_1, \alpha(\omega, G), d) = m_{P_1}(P_1, \omega, d) - 1$. Applying the inductive hypothesis in this case then gives

$$\begin{aligned} m_G(\{u, v\}, \omega, d) &\leq 1 + m_G(\{u, v\}, \alpha(\omega, G), d) \\ &\leq 1 + m_{P_1}(P_1, \alpha(\omega, G), d) \\ &= m_{P_1}(P_1, \omega, d). \end{aligned}$$

Thus we see that, for any $P \in \mathcal{P}_G(u, v)$, we have $m_G(\{u, v\}, \omega, d) \leq m_P(P, \omega, d)$. Combining this with the previous inequality, we see that

$$m_G(\{u, v\}, \omega, d) = \min_{P \in \mathcal{P}_G(u,v)} m_P(P, \omega, d),$$

as required. □

In the remainder of this chapter we prove three further results which will be useful in later chapters. The first such result is that, if P is an S-critical u-v path, where S is an optimal sequence to link u and v, then we may assume that all moves in S are played on P.

Corollary 2.5. *Let G be any connected graph with colouring ω, S an optimal sequence of moves to link $u, v \in V$ with colour d in G, and P an S-critical u-v path. Then all moves of S are played in monochromatic components intersecting P.*

Proof. First note that, by Theorem 2.4, $|S| \leq m_P(P, \omega, d)$. Let S' be the subsequence of S consisting of moves played in monochromatic components intersecting P, and without loss of generality assume that all moves in S' are in fact played on P. Then, as P is S-critical, S' played on the path P (considered as a separate graph) makes P monochromatic, and so $|S'| \geq m_P(P, \omega, d)$. But then

$$|S| \leq m_P(P, \omega, d) \leq |S'| \leq |S|,$$

so we must have equality throughout. In particular, $|S'| = |S|$ and hence all moves of S are played on P. □

Next we show that it cannot be harder to connect a pair of vertices in a larger graph.

Corollary 2.6. *Suppose G is any connected graph, with colouring ω from colour-set C, and let H be a connected subgraph of G, $d \in C$ and $u, v \in V(H)$. Then*

$$m_H(\{u, v\}, \omega, d) \geq m_G(\{u, v\}, \omega, d).$$

Proof. As H is a subgraph of G, it is clear that $\mathcal{P}_H(u,v) \subseteq \mathcal{P}_G(u,v)$. Thus, by Theorem 2.4, we have

$$m_H(\{u,v\},\omega,d) = \min_{P\in\mathcal{P}_H(u,v)} m_P(P,\omega,d) \geq \min_{P\in\mathcal{P}_G(u,v)} m_P(P,\omega,d) = m_G(\{u,v\},\omega,d).$$

□

Our final result concerns the additivity of connection times.

Corollary 2.7. *Let $G=(V,E)$ be any connected graph with colouring ω from colour-set C, and let $u,v\in V$, $xy\in E$ and $d\in C$. Then*

$$m_G(\{u,v\},\omega,d) \leq m_G(\{u,x\},\omega,d) + m_G(\{y,v\},\omega,d).$$

Proof. By Theorem 2.4, there exist u-x and v-y paths P_{ux} and P_{yv} in G such that $m_G(\{u,x\},\omega,d) = m_{P_{ux}}(P_{ux},\omega,d)$ and $m_G(\{y,v\},\omega,d) = m_{P_{yv}}(P_{yv},\omega,d)$. Then $P_{ux}P_{yv}$ gives a u-v walk in G, and so we can obtain a u-v path P_{uv} in G by deleting some vertices from $P_{ux}P_{yv}$ (and joining their neighbours). Thus

$$\begin{aligned}
m_G(\{u,v\},\omega,d) &\leq m_{P_{uv}}(P_{uv},\omega,d) && \text{by Theorem 2.4}\\
&\leq m_{P_{ux}P_{yv}}(P_{ux}P_{yv},\omega,d) && \text{by Lemma 2.1}\\
&\leq m_{P_{ux}}(P_{ux},\omega,d) + m_{P_{yv}}(P_{yv},\omega,d) && \text{by Lemma 2.2}\\
&= m_G(\{u,x\},\omega,d) + m_G(\{y,v\},\omega,d) && \text{by choice of } P_{ux}, P_{yv}.
\end{aligned}$$

□

Chapter 3

Spanning Trees and Flood-filling Games

In this chapter we investigate the relationship between the number of moves required to flood a connected graph G and the number of moves required to flood spanning trees of G. In Section 3.1, we extend ideas from the previous chapter to prove a much more powerful result: the number of moves required to flood a graph G is equal to the minimum, taken over all spanning trees T of G, of the number of moves required to flood T.

Since it remains hard to solve Free-Flood-It on trees, this result does not imply that the number of moves required to flood a graph with only a polynomial number of spanning trees can be computed in polynomial time. However, in Section 3.2 we make use of this equality to prove a number of corollaries, which we will exploit to give polynomial-time algorithms for various flood-filling problems in later chapters. Many of these corollaries generalise facts we proved for the special case of paths in the previous chapter.

3.1 The Main Theorem

In this section, we prove our characterisation of the number of moves required to flood an arbitrary graph in terms of the number of moves required to flood its spanning trees. More precisely, if $\mathcal{T}(G)$ denotes the set of all spanning trees of G, we prove the following result.

Theorem 3.1. *Let G be a connected graph with colouring ω from colour-set C. Then, for any $d \in C$,*

$$m_G(G, \omega, d) = \min_{T \in \mathcal{T}(G)} m_T(T, \omega, d).$$

To prove this theorem, we demonstrate that both the inequalities $m_G(G, \omega, d) \geq \min_{T \in \mathcal{T}(G)} m_T(T, \omega, d)$ and $m_G(G, \omega, d) \leq \min_{T \in \mathcal{T}(G)} m_T(T, \omega, d)$ must hold. We begin with an auxiliary result which is crucial to the proof of both inequalities, as it allows us to consider independently optimal sequences to flood disjoint subtrees of a tree.

Lemma 3.2. *Let T be a tree, with colouring ω from colour-set C, and let A and B be disjoint subsets of $V(T)$ such that $V(T) = A \cup B$ and $T[A], T[B]$ are connected. Then, for any $d \in C$,*

$$m_T(T, \omega, d) \leq m_{T[A]}(A, \omega, d) + m_{T[B]}(B, \omega, d).$$

Proof. First observe that the result is trivially true if either A or B is empty, so assume $A, B \neq \emptyset$. We say that a sequence of moves S played in a graph G with colouring ω *affects* the subset X of $V(G)$ if playing S in G (with colouring ω) changes the colour of at least one vertex in X at some stage. If X and Y are disjoint subsets of $V(G)$, and S_X and S_Y are sequences of moves played in X and Y respectively, we say there

is a *compatible ordering* of (X, S_X) and (Y, S_Y) if at least one of these sequences does not affect the other subset (i.e. either playing S_X in G does not change the colour of any vertex of Y, or else playing S_Y in G does not change the colour of any vertex in X).

In fact we prove the following, stronger claim.

Claim. *Let A, B and T be as above, and suppose the sequence S_A floods A with colour d, and that S_B floods B with some colour $d' \in C$. Then, if there is no compatible ordering of (A, S_A) and (B, S_B), we have*

$$m_T(T, \omega, d') \leq |S_A| + |S_B|.$$

To see that the claim implies the result, we set $d = d'$, and choose S_A and S_B to be optimal sequences to flood $T[A]$ and $T[B]$ respectively with colour d. If there is no compatible ordering of (A, S_A) and (B, S_B), the result then follows immediately from the claim (as $|S_A| + |S_B| = m_{T[A]}(A, \omega, d) + m_{T[B]}(B, \omega, d)$), so it suffices to consider the case in which there does exist a compatible ordering of (A, S_A) and (B, S_B). Without loss of generality, we may assume that playing S_A in T does not change the colour of any vertex in B.

Note that if T' is a subtree of the tree T, and S is a sequence of moves played in T', then S has exactly the same effect on T' whether the moves are played in the isolated subtree T' or in the larger tree T, as no vertices in T' can be linked in T before they are linked in T'. Thus playing S_A in T links A in colour d and (by assumption) does not change the colour of any vertex in B, so then playing S_B in the resulting coloured tree gives B colour d. If this also changes the colour of A, some vertex $b \in B$ is linked to A and so A must thereafter have the same colour as b, which ends up with colour

d. Thus the sequence $S_A S_B$ gives all of T colour d, implying

$$m_T(T, \omega, d) \leq |S_A| + |S_B| = m_{T[A]}(A, \omega, d) + m_{T[B]}(B, \omega, d),$$

as required.

We now prove the claim by induction on $|B|$. Note that we may assume without loss of generality that ω gives a proper colouring of B, otherwise we may contract monochromatic components. Suppose $|B| = 1$. If there is no compatible ordering, then S_A must change the colour of the only vertex in B (linking it to some $a \in A$), and so playing S_A in T makes the whole tree monochromatic with colour d. Thus $m_T(T, \omega, d) \leq |S_A|$, and

$$m_T(T, \omega, d') \leq m_T(T, \omega, d) + 1 \leq |S_A| + 1.$$

But the fact there is no compatible ordering also implies that $|S_B| \geq 1$, otherwise playing the empty sequence S_B in T would not affect A. Hence

$$m_T(T, \omega, d') \leq |S_A| + 1 \leq |S_A| + |S_B|,$$

as required.

Now suppose $|B| > 1$, so B is not monochromatic initially, and assume that the claim holds for smaller B. By the reasoning above, this implies that the lemma holds in these cases: in other words, whenever $\bar{T}$ is a tree and $\bar{A}$ and $\bar{B}$ are disjoint subsets of $V(\bar{T})$ such that $V(\bar{T}) = \bar{A} \cup \bar{B}$, $\bar{T}[\bar{A}]$ and $\bar{T}[\bar{B}]$ are connected, and $|\bar{B}| < |B|$, for any $\bar{d} \in C$ we have

$$m_{\bar{T}}(\bar{T}, \omega, \bar{d}) \leq m_{\bar{T}[\bar{A}]}(\bar{A}, \omega, \bar{d}) + m_{\bar{T}[\bar{B}]}(\bar{B}, \omega, \bar{d}). \tag{3.1}$$

Set ${S_B}'$ to be the initial segment of S_B, up to and including the move that first makes B monochromatic (in any colour), so any final moves that simply change the colour of B are omitted. We may, of course, have ${S_B}' = S_B$, if B is not monochromatic before the final move of S_B.

By assumption, there is no compatible ordering of (A, S_A) and (B, S_B). First suppose that there is no compatible ordering of (A, S_A) and $(B, {S_B}')$ either, and that ${S_B}'$ gives B colour $d'' \in C$ (note that this must be the situation if $S_B = {S_B}'$, in which case $d'' = d'$). Before the final move of ${S_B}'$ there are $r \geq 2$ monochromatic components in B (all but one of which have colour d''), with vertex-sets $B_1, \ldots, B_r$. For $1 \leq i \leq r$, set S_i to be the subsequence of ${S_B}'$ consisting of moves played in B_i, and note that these subsequences partition ${S_B}'$. Observe that playing S_i in B_i as an isolated graph has the same effect on the vertices of B_i as does playing ${S_B}'$ in the larger graph B: if any move played in B_i during S'_B changed the colour of any vertex outside B_i, this vertex would also have ended up in the maximal monochromatic component B_i. It follows that playing S_i in $T[B_i]$ gives B_i colour d'', so $m_{T[B_i]}(B_i, \omega, d'') \leq |S_i|$.

Let B_1 be the unique component adjacent to A, and set $T_1 = T[A \cup B_1]$. Note that S_A floods $T[A]$ with colour d, and S_1 floods $T[B_1]$ with colour d''. Moreover, as there is no compatible ordering of (A, S_A) and $(B, {S_B}')$, there cannot be a compatible ordering of (A, S_A) and (B_1, S_1): any move from ${S_B}'$ that changes the colour of a vertex in A must belong to S_1, and if S_A changes the colour of any vertex in B it must change the colour of at least one vertex in B_1. Thus we can apply the inductive hypothesis to see that

$$m_{T_1}(T_1, \omega, d'') \leq |S_A| + |S_1|.$$

Now suppose without loss of generality that B_2 is adjacent to B_1. We can then apply (3.1) to $T_2 = T[V(T_1) \cup B_2]$ to see that

$$m_{T_2}(T_2, \omega, d'') \leq m_{T_1}(T_1, \omega, d'') + m_{T[B_2]}(B_2, \omega, d'') \leq |S_A| + |S_1| + |S_2|.$$

Continuing in this way, each time adding an adjacent component, we see that

$$m_T(T, \omega, d'') \leq |S_A| + \sum_{i=1}^{r} |S_i| = |S_A| + |S_B{}'|.$$

Now, if $S_B{}' = S_B$, this immediately gives the desired result, as $d'' = d'$. Otherwise, note that $|S_B| \geq |S_B{}'| + 1$ and so

$$m_T(T, \omega, d') \leq m_T(T, \omega, d'') + 1 \leq |S_A| + |S_B{}'| + 1 \leq |S_A| + |S_B|,$$

as required.

It remains to consider the case in which there exists a compatible ordering of (A, S_A) and $(B, S_B{}')$. By assumption, there is no compatible ordering of (A, S_A) and (B, S_B), so playing S_A in T must change the colour of some vertex in B. Thus it must be that playing $S_B{}'$ in T does not change the colour of any vertex in A, although playing S_B does. Let α be the first move of S_B which, when the sequence is played in T with initial colouring ω, changes the colour of some vertex in A. Set $\tilde{S_B}$ to be the initial segment of S_B up to but not including α (so $S_B{}'$ and $\tilde{S_B}$ are both proper initial segments of S_B, with $\tilde{S_B}$ possibly longer than $S_B{}'$). Note that $\tilde{S_B}$ makes B monochromatic, and observe also that, as α is played in B but changes the colour of some vertex $a \in A$, $\tilde{S_B}$ must link a and B.

Now suppose we play $\tilde{S_B}$ in T, followed by S_A. By definition, $\tilde{S_B}$ does not change the colour of any vertex in A, so playing S_A in T with colouring $\tilde{S_B}(\omega, T)$ will make A monochromatic with colour d. However, after playing $\tilde{S_B}$, every vertex in B belongs to the same monochromatic component as $a \in A$, and so the sequence $\tilde{S_B}S_A$ must also give every vertex in B colour d. Thus we see that $m_T(T, \omega, d) \leq |\tilde{S_B}| + |S_A|$, and so

$$m_T(T, \omega, d') \leq m_T(T, \omega, d) + 1 \leq |S_A| + |\tilde{S_B}| + 1 \leq |S_A| + |S_B|,$$

completing the proof of the claim. □

This result is easily applied to give a lower bound on $m_G(G, \omega, d)$.

Lemma 3.3. *Let G be a connected graph with colouring ω from colour-set C. Then, for any $d \in C$,*

$$\min_{T \in \mathcal{T}(G)} m_T(T, \omega, d) \leq m_G(G, \omega, d).$$

Proof. It suffices to show that there exists some spanning tree T for G such that $m_T(T, \omega, d) \leq m_G(G, \omega, d)$. We prove this by induction on $m_G(G, \omega, d)$.

Note that the base case, for $m_G(G, \omega, d) = 0$, is trivial: any spanning tree will do, as all spanning trees are already monochromatic with colour d. So assume $m_G(G, \omega, d) \geq 1$ and let S be an optimal sequence to flood G with colour d.

First suppose that G is monochromatic in some colour d' before the final move of S, which simply changes the graph's colour to d, so $|S| = 1 + m_G(G, \omega, d')$. Then, by the inductive hypothesis, there exists a spanning tree T for G such that $m_T(T, \omega, d') \leq m_G(G, \omega, d')$. But we then see that

$$m_T(T, \omega, d) \leq 1 + m_T(T, \omega, d') \leq 1 + m_G(G, \omega, d') = |S| = m_G(G, \omega, d),$$

and so T is the spanning tree we require.

Thus we may assume that, immediately before the final move of S, G has $r \geq 2$ monochromatic components with vertex sets $A_1, \ldots, A_r$ (all but one of which have colour d), and that the final move is played in A_1. For $1 \leq i \leq r$, let S_i be the subsequence of S consisting of moves played in A_i; note that these sequences partition S, and that $m_{G[A_i]}(A_i, \omega, d) \leq |S_i|$ for each i (as S_i, played in $G[A_i]$, makes this subgraph monochromatic with colour d).

For $2 \leq i \leq r$, we know that $|S_i| < |S|$ (as at least one move is played in A_1), so we

may apply the inductive hypothesis immediately to see that there exists a spanning tree T_i for $G[A_i]$ such that $m_{T_i}(T_i, \omega, d) \leq m_{G[A_i]}(A_i, \omega, d) \leq |S_i|$. Now observe that, if S_1' is the subsequence S_1 with just the final move omitted, S_1' makes A_1 monochromatic with some colour d', and $|S_1'| < |S|$. Thus we can apply the inductive hypothesis here to see that there exists a spanning tree T_1 for $G[A_1]$ such that $m_{T_1}(T_1, \omega, d') \leq m_{G[A_1]}(A_1, \omega, d') \leq |S_1'|$ and so $m_{T_1}(T_1, \omega, d) \leq 1 + m_{T_1}(T_1, \omega, d') \leq 1 + |S_1'| = |S_1|$.

Note that, for $2 \leq i \leq r$, A_1 must be adjacent to A_i, and so there exists an edge e_i in G between A_1 and A_i. Define T to be $\bigcup_{i=1}^{r} T_i \cup \{e_i : 2 \leq i \leq r\}$, and observe that T is a spanning tree for G. Moreover, by Lemma 3.2 we see that

$$\begin{aligned} m_T(T, \omega, d) &\leq \sum_{i=1}^{r} m_{T_i}(T_i, \omega, d) \\ &\leq \sum_{i=1}^{r} |S_i| \\ &= |S| \\ &= m_G(G, \omega, d), \end{aligned}$$

so T is as required. □

We now proceed to show the reverse inequality. We call a spanning tree T of G *d-minimal* (with respect to the colouring ω) if $m_T(T, \omega, d) = \min_{T' \in \mathcal{T}(G)} m_{T'}(T', \omega, d)$, and say that a spanning tree T *preserves monochromatic components of* G (with respect to ω) if T and G have the same monochromatic components, i.e. $\text{comp}_G(v, \omega) = \text{comp}_T(v, \omega)$ for all $v \in V(G)$.

We shall demonstrate that, for any $d \in C$, there exists a d-minimal spanning tree T that preserves the monochromatic components of G, and that for such a tree we must have $m_T(T, \omega, d) \geq m_G(G, \omega, d)$. Our first step is to show that, given any tree T and an edge $e \notin E(T)$, we can replace T with another tree T' that contains e, without increasing the number of moves we need to flood the tree.

Lemma 3.4. *Let T be a tree with colouring ω from colour-set C, and suppose $e = uv \notin E(T)$. Then, for any $d \in C$, there exists a tree T', with $e \in E(T')$ and $E(T') \subseteq E(T) \cup \{e\}$, and which spans the vertices $V(T)$, such that $m_{T'}(T', \omega, d) \leq m_T(T, \omega, d)$.*

Proof. We proceed by induction on $m_T(T, \omega, d)$. If $m_T(T, \omega, d) = 0$ then T is already monochromatic with colour d, and we can choose T' to be any spanning tree of $T \cup \{e\}$ with $e \in E(T')$. So assume $m_T(T, \omega, d) > 0$, and let S be an optimal sequence to flood T with colour d.

First suppose that T is monochromatic in some colour d' before the final move of S, and so this last move just changes the colour to d. We can apply the inductive hypothesis to see that there exists a spanning tree T' of $T \cup \{e\}$, with $e \in E(T')$, such that $m_{T'}(T', \omega, d') \leq m_T(T, \omega, d')$. But then we have

$$m_{T'}(T', \omega, d) \leq 1 + m_{T'}(T', \omega, d') \leq 1 + m_T(T, \omega, d') = m_T(T, \omega, d).$$

Thus we may assume that the last move links $r \geq 2$ maximal monochromatic subtrees of T with vertex sets $A_1, \ldots, A_r$, and is played in A_1. Let S_i be the subsequence of S consisting of moves played in A_i, and observe that $S_1, \ldots, S_r$ partition S. Note also that, by maximality of $A_1, \ldots, A_r$, the moves of S_i (when played as part of the larger sequence S) do not change the colour of any vertices outside A_i, and so playing S in the isolated subtree $T[A_i]$ floods $T[A_i]$ with colour d, implying $m_{T[A_i]}(A_i, \omega, d) \leq |S_i|$.

Suppose first that $u, v \in A_i$ for some i. As at least one move is played in T_1, we have $m_{T[A_j]}(A_j, \omega, d) < |S| = m_T(T, \omega, d)$ for $2 \leq j \leq r$, so if $i \neq 1$ we may apply the inductive hypothesis to see that there exists a spanning tree T_i for $T[A_i] \cup \{e\}$, with $e \in E(T_i)$, such that $m_{T_i}(T_i, \omega, d) \leq m_{T[A_i]}(A_i, \omega, d) \leq |S_i|$.

For the case $i = 1$, observe that A_1 is monochromatic in some colour $d' \neq d$

before the final move of S_1, so $m_{T[A_1]}(A_1, \omega, d') < |S_1| \leq |S| = m_T(T, \omega, d)$. Thus, by the inductive hypothesis, there again exists a spanning tree T_1 for $T[A_1] \cup \{e\}$, with $e \in E(T_1)$, such that $m_{T_1}(T_1, \omega, d') \leq m_{T[A_1]}(A_1, \omega, d')$, implying

$$m_{T_1}(T_1, \omega, d) \leq 1 + m_{T_1}(T_1, \omega, d') \leq 1 + m_{T[A_1]}(A_1, \omega, d') \leq |S_1|.$$

Now let us define T' to be the tree with vertex-set $V(T)$ and edge-set $(E(T) \setminus E(T[A_i])) \cup E(T_i)$. T' clearly contains e, and by Lemma 3.2 we see that

$$\begin{aligned} m_{T'}(T', \omega, d) &\leq m_{T_i}(T_i, \omega, d) + \sum_{j \neq i} m_{T[A_j]}(A_j, \omega, d) \\ &\leq \sum_{i=1}^{r} |S_i| \\ &= |S| \\ &= m_T(T, \omega, d), \end{aligned}$$

so T' has the required properties.

It remains to consider the case that the endpoints of e do not lie in the same subtree. In particular, at most one of u and v lies in A_1, so suppose without loss of generality that $v \in A_2$. Suppose f is the unique edge in T that joins A_1 to A_2, and set T' to be the tree with vertex-set $V(T)$ and edge-set $(E(T) \setminus \{f\}) \cup \{e\}$. Once again, it is clear that T' is a tree containing e, and we can apply Lemma 3.2 to see

that

$$\begin{aligned} m_{T'}(T', \omega, d) &\leq \sum_{i=1}^{r} m_{T[A_i]}(A_i, \omega, d) \\ &\leq \sum_{i=1}^{r} |S_i| \\ &= |S| \\ &= m_T(T, \omega, d), \end{aligned}$$

completing the proof. □

Next we show that every coloured graph has a d-minimal spanning tree that preserves monochromatic components.

Lemma 3.5. *Let $G = (V, E)$ be a connected graph with colouring ω from colour-set C. Then, for any $d \in C$, there exists a d-minimal spanning tree for G that preserves monochromatic components of G with respect to ω.*

Proof. We proceed by induction on $|E|$. If $|E| = |V| - 1$ then G is a tree and the result is trivially true, so suppose $|E| \geq |V|$. If ω is a proper colouring of G then any spanning tree preserves its monochromatic components, so we may assume that there exists an edge $e \in E$ such that both endpoints of e receive the same colour under ω.

By Lemma 3.4, there exists a d-minimal spanning tree T_0 of G that contains e. Let T_1 and G_1 be the graphs obtained from T_0 and G respectively by contracting the edge e, and let ω_1 be the corresponding colouring of $V(T_1) = V(G_1)$. Note that T_1 is a spanning tree for G_1, and we have $m_{G_1}(G_1, \omega_1, d) = m_G(G, \omega, d)$ and $m_{T_1}(T_1, \omega_1, d) = m_{T_0}(T_0, \omega, d)$.

Since $|E(G_1)| < |E(G)|$, we may apply the inductive hypothesis to G_1 to find a d-minimal spanning tree T_2 of G_1 that preserves monochromatic components of G_1 with respect to ω_1. Let T be a spanning tree of G obtained from T_2 by uncontracting e (note that this tree is not necessarily unique).

It follows immediately that T preserves monochromatic components of G with respect to ω, so it remains to check that T is d-minimal. It is clear that $m_T(T, \omega, d) = m_{T_2}(T_2, \omega_1, d)$, so we see that

$$\begin{aligned} m_T(T, \omega, d) &= m_{T_2}(T_2, \omega_1, d) \\ &= \min_{T' \in \mathcal{T}(G_1)} (T', \omega_1, d) \\ &\leq m_{T_1}(T_1, \omega_1, d) \\ &= m_{T_0}(T_0, \omega, d) \\ &= \min_{T' \in \mathcal{T}(G)} (T', \omega, d), \end{aligned}$$

as required. □

We are now ready to prove our upper bound on $m_G(G, \omega, d)$.

Lemma 3.6. *Let G be a connected graph with colouring ω from colour-set C. Then, for any $d \in C$,*

$$m_G(G, \omega, d) \leq \min_{T \in \mathcal{T}(G)} m_T(T, \omega, d).$$

Proof. It suffices to prove that, for any $T \in \mathcal{T}(G)$, $m_G(G, \omega, d) \leq m_T(T, \omega, d)$. We proceed by induction on $m_T(T, \omega, d)$. If $m_T(T, \omega, d) = 0$ the result is trivially true, as G must already be monochromatic with colour d, so assume $m_T(T, \omega, d) > 0$. By Lemma 3.5, there exists a d-minimal spanning tree T' of G that preserves the monochromatic components of G with respect to ω. Note that $m_T(T, \omega, d) \geq m_{T'}(T', \omega, d)$.

Let S be an optimal sequence to flood the tree T' (considered in isolation), and let $\alpha = (v, d')$ be the first move of S. Note that $m_{T'}(T', \alpha(\omega, T'), d) < m_T(T, \omega, d)$. By choice of T', we have $\mathrm{comp}_G(v, \omega) = \mathrm{comp}_{T'}(v, \omega)$ for every $v \in V(T')$, and so

exactly the same vertices have their colour changed to d' when α is played in G as when it is played in T'. Thus $\alpha(\omega, G) = \alpha(\omega, T')$. Note that T' does not necessarily preserve monochromatic components of G with respect to the new colouring $\alpha(\omega, T')$, but we are nevertheless able to apply the inductive hypothesis to the graph with this colouring.

Thus we see that

$$\begin{aligned} m_G(G, \omega, d) &\leq 1 + m_G(G, \alpha(\omega, G), d) \\ &= 1 + m_G(G, \alpha(\omega, T'), d) \\ &\leq 1 + m_{T'}(T', \alpha(\omega, T'), d) \qquad \text{by inductive hypothesis} \\ &= m_{T'}(T', \omega, d) \\ &\leq m_T(T, \omega, d), \end{aligned}$$

as required. □

Together with the preceding results, this proves our main theorem.

Proof of Theorem 3.1. The proof follows immediately from Lemma 3.3 and Lemma 3.6. □

3.2 Corollaries

In this section, we prove five corollaries of Theorem 3.1, which will be useful in the following chapters. In the first two of these, we exploit Theorem 3.1 to generalise Lemma 3.2 very substantially. We begin by showing that the result can be extended from trees to arbitrary graphs.

Corollary 3.7. *Let G be a connected graph, with colouring ω from colour-set C, and let A and B be disjoint subsets of $V(G)$ such that $V(G) = A \cup B$ and $G[A], G[B]$ are*

connected. Then, for any $d \in C$,

$$m_G(G, \omega, d) \leq m_{G[A]}(A, \omega, d) + m_{G[B]}(B, \omega, d).$$

Proof. First note that, by Theorem 3.1, there exist spanning trees T_A and T_B of $G[A]$ and $G[B]$ respectively, such that $m_{T_A}(T_A, \omega, d) = m_{G[A]}(A, \omega, d)$ and $m_{T_B}(T_B, \omega, d) = m_{G[B]}(B, \omega, d)$. As G is connected, there exists an edge e between A and B. Set T to be the tree with vertex set $V(G)$ and edge set $E(T_A) \cup E(T_B) \cup \{e\}$. By Lemma 3.2, $m_T(T, \omega, d) \leq m_{T_A}(T_A, \omega, d) + m_{T_B}(T_B, \omega, d)$; but T is a spanning tree for G so, by Theorem 3.1,

$$\begin{aligned} m_G(G, \omega, d) &\leq m_T(T, \omega, d) \\ &\leq m_{T_A}(T_A, \omega, d) + m_{T_B}(T_B, \omega, d) \\ &= m_{G[A]}(A, \omega, d) + m_{G[B]}(B, \omega, d). \end{aligned}$$

□

We now generalise this result even further, demonstrating that we do not in fact require A and B to be disjoint.

Corollary 3.8. *Let G be a connected graph, with colouring ω from colour-set C, and let A and B be subsets of $V(G)$ such that $V(G) = A \cup B$ and $G[A], G[B]$ are connected. Then, for any $d \in C$,*

$$m_G(G, \omega, d) \leq m_{G[A]}(A, \omega, d) + m_{G[B]}(B, \omega, d).$$

Proof. If $A \cap B = \emptyset$ then the result is immediate from Corollary 3.7, so assume that

$A \cap B = X \neq \emptyset$. We now construct a new graph G' by "blowing up" every vertex in X, replacing each $v \in X$ with a pair of adjacent vertices. We set

$$V(G') = (V(G) \setminus X) \cup \{v_A, v_B : v \in X\},$$

and

$$\begin{aligned} E(G') =& E(G \setminus X) \\ & \cup \{v_A v_B : v \in X\} \\ & \cup \{v_A u, v_B u : v \in X, u \notin X, uv \in E(G)\} \\ & \cup \{v_A w_A, v_B w_B : vw \in E(G[X])\}, \end{aligned}$$

and give G' a colouring ω', where $\omega'(v_A) = \omega'(v_B) = \omega(v)$ for $v \in X$ and $\omega'(u) = \omega(u)$ for $u \notin X$. Observe that contracting monochromatic components in both (G, ω) and (G', ω') will give identical coloured graphs, so we must have

$$m_G(G, \omega, d) = m_{G'}(G', \omega', d).$$

We then define $A' = A \setminus X \cup \{v_A : v \in X\}$ and $B' = B \setminus X \cup \{v_B : v \in X\}$. Now, A' and B' partition $V(G')$ and induce connected subgraphs, so we can apply Corollary 3.7 to see that

$$m_{G'}(G', \omega', d) \leq m_{G[A']}(A', \omega', d) + m_{G[B']}(B', \omega', d).$$

But $G[A']$ with colouring ω' is identical to $G[A]$ with colouring ω, implying that $m_{G[A']}(A', \omega', d) = m_{G[A]}(A, \omega, d)$, and similarly $m_{G[B']}(G[B'], \omega', d) = m_{G[B]}(B, \omega, d)$. Thus

$$m_G(G, \omega, d) = m_{G'}(G', \omega', d) \leq m_{G[A]}(A, \omega, d) + m_{G[B]}(B, \omega, d),$$

as required. □

The next two corollaries are concerned with the number of moves required to flood a connected subgraph H of a graph G. We begin by showing that adding additional edges to H cannot increase the number of moves required to flood the graph.

Corollary 3.9. *Let G be a connected graph with colouring ω from colour-set C, and H a connected spanning subgraph of G. Then, for any $d \in C$,*

$$m_G(G, \omega, d) \leq m_H(H, \omega, d).$$

Proof. As H is a connected spanning subgraph of G, we have $\mathcal{T}(H) \subseteq \mathcal{T}(G)$. Thus, by Theorem 3.1,

$$m_G(G, \omega, d) = \min_{T \in \mathcal{T}(G)} (T, \omega, d) \leq \min_{T \in \mathcal{T}(H)} (T, \omega, d) = m_H(H, \omega, d).$$

□

Now we consider the case in which H is an arbitrary subgraph, and show that the number of moves we must play in G to link the vertices of H is at most the number required to flood the isolated subgraph H.

Corollary 3.10. *Let G be a connected graph with colouring ω from colour-set C, and H a connected subgraph of G. Then, for any $d \in C$,*

$$m_G(V(H), \omega, d) \leq m_H(H, \omega, d).$$

Proof. Set H' to be the subgraph of G induced by $\bigcup_{v \in V(H)} \mathrm{comp}_G(v, \omega)$, and note that a sequence of moves played in G floods H' if and only if it floods H (with the same

colour). Observe that we can add edges to H to obtain a coloured graph equivalent to H' (when both graphs have colouring inherited from ω): if an edge is added in H between every pair of vertices that belong to either the same monochromatic component or adjacent monochromatic components in G, then contracting monochromatic components in this new graph will give the same result as contracting monochromatic components of H'. Thus, by Corollary 3.9, $m_{H'}(H', \omega, d) \leq m_H(H, \omega, d)$, and it suffices to prove that $m_G(V(H), \omega, d) \leq m_{H'}(H', \omega, d)$.

We proceed to prove this inequality by induction on $m_{H'}(H', \omega, d)$. Note that if $m_{H'}(H', \omega, d) = 0$ then the result is trivially true, so assume $m_{H'}(H', \omega, d) > 0$. Let S be an optimal sequence to flood the (isolated) subgraph H' with colour d, and suppose the first move of S is α. As we have $\mathrm{comp}_{H'}(v, \omega) = \mathrm{comp}_G(v, \omega)$ for every $v \in V(H')$, the move α changes the colour of exactly the same vertices when played in H' as it does when played in the larger graph G. Thus $\alpha(\omega, G)|_{H'} = \alpha(\omega, H')|_{H'}$.

Note that $m_{H'}(H', \omega, d) = 1 + m_{H'}(H', \alpha(\omega, H'), d)$, so

$$m_{H'}(H', \alpha(\omega, G), d) = m_{H'}(H', \alpha(\omega, H'), d) < m_{H'}(H', \omega, d),$$

and we can apply the inductive hypothesis to see that

$$m_G(V(H'), \alpha(\omega, G), d) \leq m_{H'}(H', \alpha(\omega, G), d). \tag{3.2}$$

Hence

$$\begin{aligned} m_G(V(H),\omega,d) &= m_G(V(H'),\omega,d) \\ &\leq 1 + m_G(V(H'),\alpha(\omega,G),d) \\ &\leq 1 + m_{H'}(H',\alpha(\omega,G),d) \\ &\qquad\qquad\qquad\qquad \text{by (3.2)} \\ &= 1 + m_{H'}(H',\alpha(\omega,H'),d) \\ &\qquad\qquad\qquad\qquad \text{as } \alpha(\omega,G)|_{H'} = \alpha(\omega,H')|_{H'} \\ &= m_{H'}(H',\omega,d), \end{aligned}$$

as required. □

Finally, we consider the number of moves required to connect a given subset of the vertices of G. For any $U \subseteq V(G)$, let $\mathcal{T}(U,G)$ be the set of all subtrees T of G such that $U \subseteq V(T)$. We then characterise the number of moves required to link U in terms of the number of moves required to flood elements of $\mathcal{T}(U,G)$.

Corollary 3.11. *Let G be a connected graph with colouring ω from colour-set C, and let $U \subseteq V(G)$. Then, for any $d \in C$,*

$$m_G(U,\omega,d) = \min_{T \in \mathcal{T}(U,G)} m_T(T,\omega,d).$$

Proof. We begin by showing that $m_G(U,\omega,d) \leq \min_{T\in\mathcal{T}(U,G)} m_T(T,\omega,d)$. Let $T \in \mathcal{T}(U,G)$. Then, as $U \subseteq V(T)$, we see by definition of $m_G(U,\omega,d)$ that $m_G(U,\omega,d) \leq m_G(V(T),\omega,d)$. Moreover, by Corollary 3.10, $m_G(V(T),\omega,d) \leq m_T(T,\omega,d)$, and so

$m_G(U, \omega, d) \leq m_T(T, \omega, d)$. As this holds for any $T \in \mathcal{T}(U, G)$, it follows that

$$m_G(U, \omega, d) \leq \min_{T \in \mathcal{T}(U,G)} m_T(T, \omega, d).$$

To show the reverse inequality, suppose that S is an optimal sequence to link U in G. Let A be the vertex set of the monochromatic component, with respect to $S(\omega, G)$, that contains U, and let S' be the subsequence of S consisting of moves played in A. Then S' makes $G[A]$ monochromatic with colour d, and so we have

$$m_G(U, \omega, d) = |S| \geq |S'| \geq m_{G[A]}(A, \omega, d).$$

By Theorem 3.1, we know that there exists a spanning tree T_A for A such that $m_A(A, \omega, d) = m_{T_A}(T_A, \omega, d)$. But then $U \subseteq V(T)$, so $T_A \in \mathcal{T}(U, G)$ and we have

$$m_G(U, \omega, d) \geq m_{G[A]}(A, \omega, d) = m_{T_A}(T_A, \omega, d) \geq \min_{T \in \mathcal{T}(U,G)} m_T(T, \omega, d).$$

□

In summary, we see from Corollary 3.8 that the number of moves required to flood a graph is bounded above by the sum of the numbers of moves required to flood connected subgraphs with cover the vertex set, whereas Corollaries 3.9 and 3.10 show that adding edges and vertices to a connected graph H can only decrease the number of moves required to make the vertex-set of H monochromatic. Corollary 3.11 allows us to calculate the minimum number of moves required to connect some subset of the vertices by considering the number of moves required to flood subtrees of G.

Chapter 4

Efficient Algorithms for Flood-filling Problems

In this chapter, we make use of results from Chapters 2 and 3 to give polynomial-time algorithms for a number of flood-filling problems. We begin in Section 4.1 by showing that 2-Free-Flood-It is solvable in polynomial time on arbitrary graphs (a result proved independently by Lagoutte [40] and Clifford et. al. [16]). Then in Section 4.2 we prove that both Fixed-Flood-It and Free-Flood-It can be solved in polynomial time on graphs that have only a polynomial number of connected subgraphs.

The remainder of the chapter focuses on the problem k-Linking-Flood-It, in which the goal is to determine the number of moves required to link an arbitrary subset of k vertices. First, in Section 4.3.1, we show that 2-Linking-Flood-It is solvable in polynomial time on arbitrary graphs; an easy corollary of this result is that Free-Flood-It is polynomially solvable on paths. Finally, in Section 4.3.2, we show more generally that k-Linking-Flood-It can be solved in polynomial time on arbitrary graphs, for any fixed k.

4.1 The Complexity of 2-FREE-FLOOD-IT on arbitrary graphs

In this section we consider the free version of two-colour Flood-It, played on an arbitrary connected graph $G = (V, E)$. We may assume that the initial two-colouring is proper, as we may otherwise contract monochromatic components to obtain an equivalent coloured graph. When making a move $m = (v, d)$ in such a game, our only choice is the vertex at which we play, as there is only one possible way to change its colour (note therefore that the fixed version becomes trivial with two colours). Making a move in the game then corresponds to picking a vertex $v \in V$ and contracting all edges incident with it, and the aim of the game is to reduce the graph to a single vertex using as few moves as possible. We can then regard any strategy as a sequence of vertices around which we perform contractions. Of course, we may contract at a vertex w which was created by an earlier contraction, but in this case we can always choose a vertex u from the original graph as a representative for w, and regard the contraction as being performed about u.

In the following lemma, we make the key observation that we may assume all contractions are performed around a single vertex.

Lemma 4.1. *There exists an optimal strategy in which we contract at the same vertex in every move.*

Proof. Suppose that, for some $v_1, \ldots, v_k \in V$, $S = v_1^{r_1} \ldots v_k^{r_k}$ is an optimal sequence, with k as small as possible (where we perform r_i consecutive contractions about the vertex v_i, so optimality implies that $\sum_{i=1}^{k} r_i$ is as small as possible). We will show that, if $k \geq 2$, there exists a sequence of moves, of no greater length, which contracts the graph to a single vertex and in which contractions are performed about only $k-1$ distinct vertices, contradicting the minimality of k and thus proving the result.

Let us denote by G' the graph obtained by performing the sequence of contractions

$v_1^{r_1} \dots v_{k-2}^{r_{k-2}}$, so the remaining contractions about v_{k-1} and v_k reduce G' to a single vertex. We claim that there exists a single vertex w such that all vertices in G' are at distance at most $r_{k-1} + r_k$ from w, and hence we can perform $r_{k-1} + r_k$ contractions about w to reduce G' to a single vertex, giving our contradiction to the minimality of k.

Consider a shortest path P from v_{k-1} to v_k in G'. Without loss of generality we may assume $d(v_{k-1}, v_k) > r_{k-1}$, otherwise v_k is absorbed by the contractions performed around v_{k-1} and, in order to minimise the number of distinct vertices, we would have chosen v_{k-1} as a representative for the vertex about which we perform the remaining contractions. Observe also that the length of P is at most $r_{k-1} + r_k$, or the two final sets of contractions would not reduce P to a single vertex.

Let $\alpha = r_{k-1} + r_k - d(v_{k-1}, v_k) \geq 0$. We can then consider the last $r_{k-1} + r_k$ moves of S in three stages.

1. The first r_{k-1} moves contract all vertices at distance at most r_{k-1} from v_{k-1} in G' to a single vertex, u_1, in the new graph G_1.

2. The next $r_k - \alpha$ moves contract all vertices at distance at most $r_k - \alpha$ from v_k in G_1 to a single vertex u_2 in the new graph G_2. Note that u_1 is absorbed only at the final step.

3. The remaining α moves absorb only vertices within distance α of u_2 in G_2. Thus we absorb any vertices at distance at most r_k from v_k in G', and additionally any other vertices at distance at most α from u_1 in G_1, that is vertices at distance at most $r_{k-1} + \alpha$ from v_{k-1} in G'.

Hence, as these $r_{k-1} + r_k$ moves reduce G' to a single vertex, we know that for every vertex $x \in G'$, either $d(x, v_k) \leq r_k$, or $d(x, v_{k-1}) \leq r_{k-1} + \alpha$.

Now set w to be the vertex on P at distance r_{k-1} from v_k. It remains to check that if $d(x, v_k) \leq r_k$ or $d(x, v_{k-1}) \leq r_{k-1} + \alpha$ then we have $d(x, w) \leq r_{k-1} + r_k$.

First suppose $d(x, v_k) \leq r_k$. Then

$$d(x, w) \leq d(x, v_k) + d(v_k, w) \leq r_k + r_{k-1},$$

as required. Now suppose that $d(x, v_{k-1}) \leq r_{k-1} + \alpha$. But then we have

$$\begin{aligned} d(x, w) &\leq d(x, v_{k-1}) + d(v_{k-1}, w) \\ &\leq r_{k-1} + \alpha + d(v_{k-1}, v_k) - r_{k-1} \\ &= r_{k-1} + r_k, \end{aligned}$$

as required. □

Theorem 4.2. *2-*Free-Flood-It *is solvable in polynomial time on arbitrary graphs.*

Proof. By Lemma 4.1, it is enough to consider strategies in which we contract about the same vertex in every move. It is clear that the number of moves required, if we always contract around the vertex v, is equal to $\max_{u \in V(G)} d(u, v)$, and that the minimum number of moves required to flood the entire graph is obtained by taking the minimum over all possible vertices v. But this is exactly equal to the radius of the graph, which can easily be computed in polynomial time. □

4.2 Graphs with Polynomial Bounds on the Number of Connected Subgraphs

Given a vertex v in an arbitrary graph G, the number of possible values of $\mathrm{comp}_G(v, \omega)$, as ω ranges over all possible colourings of G, will in general be exponential. However, it is clear that $\mathrm{comp}_G(v, \omega)$ must be a connected subgraph of G containing v, and in some interesting classes of graphs the number of connected subgraphs containing any

given vertex is bounded by a polynomial function of $|G|$. In this section we discuss polynomial time algorithms to solve flood-filling problems in this situation.

First, in Section 4.2.1, we apply corollaries of Theorem 3.1 to show that Free-Flood-It can be solved in polynomial time on graphs which have only a polynomial number of connected subgraphs. Then, in Section 4.2.2, we give a direct proof that the same is true for the fixed variant.

A graph G is said to be a subdivision of another graph H if G can be obtained from H by repeatedly subdividing edges, where subdividing the edge uv means to replace uv by the the path $uv', v'v$ for some new vertex v'. It is easy to check that, if H is a fixed graph (so we can take $|H|$ to be constant) and G is any subdivision of H, the number of connected subgraphs of G is bounded by a polynomial function of $|G|$, and so our results here imply that Free-Flood-It is solvable in polynomial time on subdivisions of any fixed graph H (recall that, for any graph or subgraph H, $|H|$ denotes the number of vertices in H). This is a stronger result than the conjecture of Meeks and Scott [44] that c-Free-Flood-It is solvable in polynomial time on subdivisions of any fixed graph H, and also substantially extends a result of Fukui, Nakanishi, Uehara, Uno and Uno [27] that Free-Flood-It is polynomially solvable on cycles.

It should be noted, however, that this condition of having only a polynomial number of connected subgraphs is not *necessary* for a graph to admit a polynomial-time algorithm to solve Free-Flood-It. K_n has $\Theta(2^n)$ connected induced subgraphs, but the number of moves required to flood the graph in either version of the game is always one fewer than the number of colours used in the initial colouring. Graphs corresponding to rectangular $2 \times n$ boards give another such example for the fixed case, as there are $\Omega(2^n)$ connected subgraphs containing any given vertex but Fixed-Flood-It can be solved in linear time in this setting [16].

4.2.1 The Free-Flood-It case

In this section we prove the following theorem.

Theorem 4.3. *Let p be a polynomial, and let $\mathcal{G}_p$ be the class of graphs such that, for any $G \in \mathcal{G}_p$, the number of connected subgraphs of G is at most $p(|G|)$. Suppose $G \in \mathcal{G}_p$ has colouring ω from colour-set C. Then, for any $d \in C$, we can compute $m_G(G, \omega, d)$ in polynomial time, and hence we can also compute $m_G(G, \omega)$ in polynomial time.*

In the following theorem, we give an explicit bound on the time taken to solve Free-Flood-It in terms of the number of connected subgraphs in the graph we are considering; Theorem 4.3 follows immediately from this result. The proof relies on Corollary 3.8, which allows us to consider optimal sequences in distinct components of the graph independently.

Theorem 4.4. *Let G be a connected graph with colouring ω from colour-set C, and suppose G has at most N connected subgraphs. Then we can compute $m_G(G, \omega, d)$ for every $d \in C$, and hence $m_G(G, \omega)$, in time $O(|C|^2 \cdot N^2)$.*

Proof. Note that we may assume without loss of generality that ω is a proper colouring of G, otherwise we can contract monochromatic components to obtain an equivalent coloured graph. Let $\mathcal{H}$ be the set of connected subgraphs of G. We compute $m_H(H, \omega, d_1)$ recursively, for each $H \in \mathcal{H}$ and $d_1 \in C$. For any $H \in \mathcal{H}$ we write $(A, B) \in \text{split}(H)$ if A and B are connected proper subgraphs of H such that $V(A) \cup V(B) = V(H)$ and $V(A) \cap V(B) = \emptyset$.

We define a function $m^*(H, \omega, d_1)$, and claim that for any $H \in \mathcal{H}$ and $d_1 \in C$, we have $m_H(H, \omega, d_1) = m^*(H, \omega, d_1)$. We first define

$$m^*(\{v\}, \omega, d_1) = \begin{cases} 0 & \text{if } \omega(v) = d_1 \\ 1 & \text{otherwise.} \end{cases}$$

and observe that this gives $m_H(H, \omega, d_1) = m^*(H, \omega, d_1)$ whenever $|H| = 1$. Further values of m^* are defined recursively as follows:

$$\begin{aligned} m^*(H, \omega, d_1) = \\ \min\{ &\min_{(A,B)\in \mathrm{split}(H)} \{m_A(A, \omega, d_1) + m_B(B, \omega, d_1)\}, \\ &1 + \min_{\substack{(A,B)\in \mathrm{split}(H) \\ d_2 \in C}} \{m_A(A, \omega, d_2) + m_B(B, \omega, d_2)\}\}, \end{aligned} \tag{4.1}$$

The fact that $m_H(H, \omega, d_1) \leq m^*(H, \omega, d_1)$ follows from Corollary 3.7. Observe that, by Corollary 3.7, we have $m_H(H, \omega, d) \leq m_A(A, \omega, d) + m_B(B, \omega, d)$ for any A and B such that $(A, B) \in \mathrm{split}(H)$. Since $m_H(H, \omega, d) \leq 1 + m_H(H, \omega, d')$ for any $d' \in C$ (if H is monochromatic, we can change its colour to d in a single move), we also see by Corollary 3.7 that $m_H(H, \omega, d) \leq 1 + m_A(A, \omega, d') + m_B(B, \omega, d')$ for any $d' \in C$ and A, B with $(A, B) \in \mathrm{split}(H)$. Taking the minimum over all such possible values of A, B and d', we have $m_H(H, \omega, d) \leq m^*(H, \omega, d)$, as required.

To see the reverse inequality in the case that $|H| > 1$ (and so by assumption H is not monochromatic under ω), we consider the final move α in an optimal sequence to flood H with colour d_1: either α changes the colour of some monochromatic component X, linking it to monochromatic components $Y_1, \ldots, Y_r$ which already have colour d_1, or else H is already monochromatic in some colour d_2 before the final move, and α simply changes its colour to d_1. In the first case, we set $A = Y_1$ and $B = X \cup Y_2 \cup \ldots \cup Y_r$, and note that the disjoint subsequences of S consisting of moves played in A and B respectively flood the relevant subgraphs with colour d_1. Hence $|S| \geq m_A(A, \omega, d_1) + m_B(B, \omega, d_1)$. In the case that H is monochromatic before α, we observe that H cannot be monochromatic before the penultimate move of S (otherwise S would not be optimal) and apply the reasoning above to the initial segment S' of S in which the final move is omitted, a sequence which floods H with colour d_2: there exists $(A, B) \in \mathrm{split}(H)$ such that $|S'| \geq m_A(A, \omega, d_2) + m_B(B, \omega, d_2)$,

and hence $|S| \geq 1 + m_A(A, \omega, d_2) + m_B(B, \omega, d_2)$. Thus in either case we have $m^*(H, \omega, d_1) \leq m_H(H, \omega, d_1)$.

Observe that every subgraph on the right hand side of (4.1) contains strictly fewer vertices than H, and so a recursion based on this relationship will terminate. Thus it remains to show that we can calculate $m^*(H, \omega, d_1)$ for all $H \in \mathcal{H}$ and $d_1 \in C$ in time $O(|C|^2 \cdot N^2)$.

First we need to construct a list of all connected subgraphs of G. Clearly each vertex in the graph is a connected subgraph of order one, and given all connected subgraphs of order k we can construct all connected subgraphs of order $k+1$ by considering all possible ways of adding a vertex. Thus, if N_i denotes the number of connected subgraphs of order i in G, we can construct the list in time

$$n + \sum_{i=1}^{n-1} N_i(n-i) \leq n \cdot N = O(N^2).$$

To compute m^*, we begin by initialising the table in time $O(|G|)$, then all further values of m^* are then calculated using (4.1). This involves taking the minimum over at most $|N| \cdot |C|$ values, as the choice of subgraph A determines the subgraph B, so there is at most one value for each combination of a connected subgraph of G and colour from C. Thus we can compute the values of all entries in the table in time at most $O(N^2 \cdot |C|^2)$. This immediately gives $m_G(G, \omega, d_1)$ for each $d_1 \in C$, and to compute $m_G(G, \omega)$ we simply take the minimum over $|C|$ entries. Thus we can compute both $m_G(G, \omega, d)$ and $m_G(G, \omega)$ in time $O(N^2 \cdot |C|^2)$. □

4.2.2 The Fixed-Flood-It case

In this section we show that the fixed variant, Fixed-Flood-It, can be solved in polynomial time on any coloured graph (G, ω) in which there are a polynomial number of connected subgraphs. As a special case, this gives an alternative proof of Lagoutte's result [40] that Fixed-Flood-It $\in \mathbf{P}$ when restricted to cycles, as a cycle has only a quadratic number of connected subgraphs.

When considering the fixed variant of the game, we use the same notation as before but add a superscript to denote the fixed root vertex at which we play, writing for example $m_G^{(v)}(G, \omega, d)$ for the minimum number of moves that must be played at v to give all vertices colour d.

Theorem 4.5. *Let p be a polynomial, and let $\mathcal{G}_p$ be the class of graphs such that, for any $G \in \mathcal{G}_p$, the number of connected subgraphs of G is at most $p(|G|)$. Suppose $G \in \mathcal{G}_p$ has colouring ω from colour-set C. Then, for any $d \in C$ and $v \in V(G)$, we can compute $m_G^{(v)}(G, \omega, d)$ in polynomial time, and hence we can also compute $m_G^{(v)}(G, \omega)$ in polynomial time.*

Proof. We define a set of *states* $\mathcal{S}$, where each $S_i \in \mathcal{S}$ is a pair (A_i, d_i) with $d_i \in C$, and A_i a connected subgraph of G containing v; we say we are in state S_i if A_i is the maximal monochromatic component containing v and has colour d_i. We now construct a digraph D with vertex-set $\mathcal{S}$ and edge set E, where $(A_i, d_i)(A_j, d_j) \in E$ if and only if $A_i \subseteq A_j$, $A_j \setminus A_i$ is either empty or has colour d_j under ω, and no vertex in $\Gamma(A_j) \setminus A_j$ has colour d_j. Thus there is a directed edge from S_i to S_j if and only if we can reach state S_j from state S_i with a single move. Note that we can construct D in time polynomial in $|\mathcal{S}|$.

Let us denote by S_0 the initial state (so the tuple S_0 consists of the maximal monochromatic component containing v at the start, and its initial colour under ω), and by S_d the state (G, d) in which the entire graph is monochromatic with colour d.

Then the problem of computing $m^{(v)}(G, \omega, d)$ is exactly that of finding the length of a shortest path from S_0 to S_d in D, which can be done for all $d \in C$ in time $O(|\mathcal{S}|^2)$ (by Dijkstra's algorithm; see [17]). Thus we can construct D and compute $m^{(v)}(G, \omega, d)$ in time bounded by a polynomial in $|\mathcal{S}|$; since $|\mathcal{S}| \leq p(|G|) \cdot |C| \leq |G|^{O(1)}$ (as $|C| \leq |G|$), it follows that we can do all of this in time polynomial in $|G|$. To calculate $m^{(v)}(G, \omega)$ we simply have to take the minimum over $|C|$ values from this computation, so we can also calculate $m^{(v)}(G, \omega)$ in polynomial time. □

4.3 Connecting Subsets of Vertices

4.3.1 The complexity of 2-Linking-Flood-It

In this section, we consider how to calculate the minimum number of moves required to link each pair of vertices in an arbitrary graph. This problem is similar to the *all-pairs shortest path* problem, which can be solved in time $O(|V|^3)$ using the Floyd-Warshall algorithm, as described in [17]. Here, however, the situation is somewhat more complex: firstly, we need to consider the different costs associated with linking pairs of vertices in different colours, and secondly we cannot simply add costs when we concatenate paths. These factors lead to the greater complexity of our algorithm. The proof relies heavily on results proved in Chapter 2, in particular the characterisation of the minimum number of moves required to link two vertices u and v in the graph G in terms of the number of moves required to flood u-v paths in G.

Theorem 4.6. *Let $G = (V, E)$ be a connected graph with colouring ω from colour-set C. Then we can compute $m_G(\{u, v\}, \omega, d)$ for every pair $(u, v) \in V^{(2)}$ and every $d \in C$ in time $O(|V|^3|E||C|^2)$.*

Proof. We begin by defining a function $m^*(u, v, d, i)$ for each $(u, v) \in V^{(2)}$, $d \in C$,

and index $1 \leq i \leq |V|$. For $i = 1$, we set

$$m^*(u, v, d, 1) = \begin{cases} 0 & \text{if } u = v \text{ and } \omega(v) = d, \\ 1 & \text{if } u = v \text{ and } \omega(v) \neq d, \\ \infty & \text{otherwise.} \end{cases}$$

Observe that, for all $v \in V$ and $d \in C$, if $u = v$ then $m^*(u, v, d, 1) = m_G(\{u, v\}, \omega, d)$. We define further values of m^* recursively, setting, for each $u, v \in V$, $d \in C$, and $2 \leq i \leq |V|$,

$$m^*(u, v, d, i) = \min_{\substack{xx' \in E \\ j_1, j_2 \geq 1, j_1 + j_2 \leq i}} \{m^*(u, x, d, j_1) + m^*(x', v, d, j_2), \min_{d' \in C}\{1 + m^*(u, x, d', j_1) + m^*(x', v, d', j_2)\}\}. \quad (4.2)$$

Note that all occurrences of m^* on the right-hand side of this equation have an index strictly smaller than i, so a recursion based on this relationship will terminate.

We claim that, for all $u, v \in V(G)$ and $d \in C$,

$$m^*(u, v, d, |V|) = m_G(\{u, v\}, \omega, d).$$

To prove this claim, we first show by induction on i that we always have $m^*(u, v, d, i) \geq m_G(\{u, v\}, \omega, d)$. Note that the base case, for $i = 1$, follows immediately from the definitions above. Assume therefore that $i > 1$ and that the result holds for all smaller values of i.

Let xx' be any edge of G, and $j_1, j_2 \geq 1$ any pair of integers such that $j_1 + j_2 \leq i$. Then, as $j_1, j_2 < i$, we can apply the inductive hypothesis to see that $m^*(u, x, d, j_1) \geq m_G(\{u, x\}, \omega, d)$ and $m^*(x', v, d, j_2) \geq m_G(\{x', v\}, \omega, d)$. Similarly, for any $d' \in C$, $m^*(u, x, d', j_1) \geq m_G(\{u, x\}, \omega, d')$ and $m^*(x', v, d', j_2) \geq m_G(\{x', v\}, \omega, d')$. We can

then apply Corollary 2.7 to see that

$$\begin{aligned} m^*(u,x,d,j_1) + m^*(x',v,d,j_2) &\geq m_G(\{u,x\},\omega,d) + m_G(\{x',v\},\omega,d) \\ &\geq m_G(\{u,v\},\omega,d). \end{aligned}$$

Corollary 2.7 also gives

$$m^*(u,x,d',j_1) + m^*(x',v,d',j_2) \geq m_G(\{u,v\},\omega,d'),$$

and so, as the colour in which two vertices are linked can be changed with a single move,

$$1 + m^*(u,x,d',j_1) + m^*(x',v,d',j_2) \geq m_G(\{u,v\},\omega,d).$$

Thus we see that every term on the right-hand side of (4.2) is at least $m_G(\{u,v\},\omega,d)$, and so $m^*(u,v,d,i) \geq m_G(\{u,v\},\omega,d)$, as required.

To complete the proof of the claim, it remains to show that $m^*(u,v,d,|V|) \leq m_G(\{u,v\},\omega,d)$. To do this, we demonstrate that, for any i,

$$m^*(u,v,d,i) \leq \min_{\substack{P \in \mathcal{P}_G(u,v) \\ |P| \leq i}} m_G(P,\omega,d). \tag{4.3}$$

Since any path in G has at most $|V|$ vertices, this will imply that

$$m^*(u,v,d,|V|) \leq \min_{P \in \mathcal{P}_G(u,v)} m_G(P,\omega,d),$$

and hence, by Theorem 2.4,

$$m^*(u,v,d,|V|) \leq m_G(\{u,v\},\omega,d).$$

To prove the validity of (4.3), we again proceed by induction on i. Observe that

the claim is true in the base case, for $i = 1$, so assume that $i > 1$ and that the claim holds for all smaller values of i. Let $P \in \mathcal{P}_G(u, v)$ be any path such that $|P| \leq i$, and let S be an optimal sequence of moves to make the isolated path P (with colouring ω) monochromatic with colour d (so $|S| = m_G(P, \omega, d)$). Our goal is then to prove that $m^*(u, v, d, i) \leq |S|$. We now proceed by case analysis on the final move of S; there are three possibilities.

1. The last move links two monochromatic segments of P. Without loss of generality, suppose $xx' \in E(P)$ is such that the segment P_1 from u to x has colour d, and the segment P_2 from x' to v has colour d'. Note that $P_1 \in \mathcal{P}_G(u, x)$ and $P_2 \in \mathcal{P}_G(x', v)$. We then have

$$\begin{aligned}
|S| &\geq 1 + m_{P_1}(P_1, \omega, d) + m_{P_2}(P_2, \omega, d') \\
&\geq m_{P_1}(P_1, \omega, d) + m_{P_2}(P_2, \omega, d) \\
&\geq \min_{\substack{P' \in \mathcal{P}_G(u,x) \\ |P'| \leq |P_1|}} m_{P'}(P', \omega, d) + \min_{\substack{P'' \in \mathcal{P}_G(x',v) \\ |P''| \leq |P_2|}} m_{P''}(P'', \omega, d) \\
&\geq m^*(u, x, d, |P_1|) + m^*(x', v, d, |P_2|) \\
&\qquad\qquad \text{by inductive hypothesis, as } |P_1|, |P_2| < i \\
&\geq m^*(u, v, d, i) \\
&\qquad\qquad \text{by (4.2), as } xx' \in E(G) \text{ and } |P_1| + |P_2| = |P| \leq i.
\end{aligned}$$

2. The last move links three monochromatic segments of P, of which the end two must already have colour d. Suppose $xx', yy' \in E(P)$ are such that the segment P_1 from u to x has colour d, the segment P_2 from x' to y has colour d', and the final segment P_3 from y' to v has colour d. Note that $P_1 \in \mathcal{P}_G(u, x)$,

$P_2 \in \mathcal{P}_G(x', y)$ and $P_3 \in \mathcal{P}_G(y', v)$. We then have

$$\begin{aligned}
|S| &\geq 1 + m_{P_1}(P_1, \omega, d) + m_{P_2}(P_2, \omega, d') + m_{P_3}(P_3, \omega, d) \\
&\geq m_{P_1}(P_1, \omega, d) + m_{P_2}(P_2, \omega, d) + m_{P_3}(P_3, \omega, d) \\
&\geq \min_{\substack{P' \in \mathcal{P}_G(u,x) \\ |P'| \leq |P_1|}} m_{P'}(P', \omega, d) + \min_{\substack{P'' \in \mathcal{P}_G(x',y) \\ |P''| \leq |P_2|}} m_{P''}(P'', \omega, d) \\
&\qquad + \min_{\substack{P''' \in \mathcal{P}_G(y',v) \\ |P'''| \leq |P_3|}} m_{P'''}(P''', \omega, d) \\
&\geq m^*(u, x, d, |P_1|) + m^*(x', y, d, |P_2|) + m^*(y', v, d, |P_3|) \\
&\qquad \text{by inductive hypothesis, as } |P_1|, |P_2|, |P_3| < i \\
&\geq m^*(u, x, d, |P_1|) + m^*(x', v, d, |P_2| + |P_3|) \\
&\qquad \text{by (4.2), as } yy' \in E(G) \\
&\geq m^*(u, v, d, i) \\
&\qquad \text{by (4.2), as } xx' \in E(G) \text{ and } |P_1| + |P_2| + |P_3| = |P| \leq i.
\end{aligned}$$

3. P is already monochromatic before the last move, which simply changes its colour from some $d' \in C$ to d. By optimality of S, the path cannot be monochromatic before the penultimate move. Thus, if we set S' to be the subsequence of S with just the final move omitted, we can see by the previous two cases that there must exist $xx' \in E(G)$ and integers $j_1, j_2 \geq 1$ with $j_1 + j_2 = |P|$ such that

$$|S'| \geq m^*(u, x, d', j_1) + m^*(x', v, d', j_2),$$

and hence

$$|S| = 1 + |S'| \geq 1 + m^*(u, x, d', j_1) + m^*(x', v, d', j_2).$$

Since $j_1 + j_2 = |P| \leq i$, it follows from (4.2) that $|S| \geq m^*(u, v, d, i)$.

So in all cases we have $|S| \geq m^*(u, v, d, i)$, completing the proof that (4.3) holds, and hence the proof that $m^*(u, v, d, |V|) = m_G(\{u, v\}, \omega, d)$ for all values of $u, v \in V$ and $d \in C$. Thus it suffices to compute all values of m^*, which we do using a dynamic programming technique.

In our dynamic program, the table has an entry for each pair of vertices $u, v \in V$, each colour $d \in C$, and each possible value of the index i, giving a total of $|V|^3 \cdot |C|$ entries. We initialise each entry for which $i = 1$ in constant time (to either 0, 1 or ∞), and all other entries depend only on those where the value of the index i is strictly smaller. To compute the value of an entry with $i > 1$, we need to take the minimum over at most $|E| \cdot |C|$ combinations of pairs of existing values in the table. Thus the entire computation can be performed in time $O(|V|^3|E||C|^2)$. □

We obtain an easy corollary by applying this result to the special case in which the graph in question is a path.

Corollary 4.7. Free-Flood-It *can be solved for any path P in time $O(|P|^6)$, and* c-Free-Flood-It *can be solved in time $O(|P|^4)$.*

Proof. Let P be a path with colouring ω, and let u and v be the two end-vertices of P. Then a sequence of moves S makes P monochromatic if and only if it links u and v, so $m(P, \omega) = m_P(\{u, v\}, \omega)$. But by Theorem 4.6, we can compute $m_G(\{u, v\}, \omega)$ for two vertices in any arbitrary graph G with initial colouring ω in time $O(|V|^3|E||C|^2)$. As P has $O(|P|)$ edges, and we cannot possibly have a colour-set of size greater than $|P|$, the complexity of this algorithm is bounded by $O(|P|^6)$, or $O(|P|^4)$ if the colour-set has fixed size. □

4.3.2 The complexity of k-Linking-Flood-It

In this section we generalise substantially the main result of the previous section, proving that k-Linking-Flood-It, the problem of determining the minimum num-

ber of moves required to link some given set of k points (when moves can be played at any vertex), is solvable in polynomial time for any fixed k. As a special case, this gives a polynomial-time algorithm for the 2-Linking-Flood-It problem considered in the previous section; however, the running time of this more general algorithm with $k = 2$ is slightly worse than the algorithm previously described.

We begin with some additional notation. Let U be a subset of $V(G)$. We will say $(U_1, U_2) \in \text{part}(U)$ if U_1 and U_2 are disjoint nonempty subsets of U such that $U = U_1 \cup U_2$. Recall that $\mathcal{T}(U, G)$ is the set of all subtrees T of G such that $U \subseteq V(T)$. For $1 \leq i \leq |G|$, set $\mathcal{T}_i(U, G) = \{T \in \mathcal{T}(U, G) : |T| \leq i\}$.

Recall from Corollary 3.11 that, for any $U \subseteq V(G)$,

$$m_G(U, \omega, d) = \min_{T \in \mathcal{T}(U,G)} m_T(T, \omega, d).$$

We use this result to give a dynamic programming algorithm to solve k-Linking-Flood-It in polynomial time, for any fixed k. Note that the statement of the theorem assumes that the initial colouring of the graph being considered is proper, but of course if this is not the case we can simply contract monochromatic components to obtain an equivalent properly coloured graph.

Theorem 4.8. *Let $G = (V, E)$ be a connected graph of order n, with proper colouring ω from colour-set C, and let $U \subseteq V$ with $|U| = k$. Then, for any $d \in C$, we can compute $m_G(U, \omega, d)$ in time $O(n^{k+3} \cdot |E| \cdot |C|^2 \cdot 2^k)$.*

Proof. We demonstrate a dynamic programming algorithm to compute values of a function f, taking as arguments a nonempty subset $W \subset V$ of at most k vertices, the initial colouring ω of the graph, a colour $d_1 \in C$, and an index $i \in \{1, \ldots, n\}$. We

will show that, for any values of these arguments, we have

$$f(W,\omega,d_1,i)=\begin{cases}\min_{T\in\mathcal{T}_i(W,G)} m_T(T,\omega,d_1) & \text{if } \mathcal{T}_i(W,G)\neq\emptyset\\ \infty & \text{otherwise.}\end{cases}$$

Thus, as $\mathcal{T}_n(U,G)\neq\emptyset$, we see by Corollary 3.11 that

$$m_G(U,\omega,d)=\min_{T\in\mathcal{T}(U,G)} m_T(T,\omega,d)=\min_{T\in\mathcal{T}_n(U,G)} m_T(T,\omega,d)=f(U,\omega,d,n).$$

We initialise our table by setting

$$f(W,\omega,d_1,1)=\begin{cases}\infty & \text{if } |W|\geq 2\\ 1 & \text{if } W=\{w\} \text{ and } \omega(w)\neq d_1\\ 0 & \text{if } W=\{w\} \text{ and } \omega(w)=d_1,\end{cases}$$

and observe that this gives the desired value of $f(W,\omega,d_1,1)$ for all choices of W and d_1.

We define further values of f recursively. First, for any W, ω, d_1 and i, we set

$$\begin{aligned}\mathrm{poss}(W,\omega,d_1,i)=\{&((W_1\cup\{x_1\},\omega,d_1,j_1),(W_2\cup\{x_2\},\omega,d_1,j_2)):\\ &(W_1,W_2)\in\mathrm{part}(W),x_1x_2\in E,x_1\notin W_2,x_2\notin W_1,\\ &j_1+j_2=i,j_1,j_2>0\},\end{aligned}$$

so there is an element of $\mathrm{poss}(W,\omega,d_1,i)$ corresponding to each way of partitioning W into two non-empty subsets, each way of picking an edge in G and associating one endpoint with each subset, and each pair of positive integers summing to i. Note that we may or may not have $x_i\in W_i$; the need for this flexibility will become clear on

page 63.

We then define, for $i \geq 2$,

$$f_1(W, \omega, d_1, i) = \begin{cases} \min_{(\mathbf{z_1}, \mathbf{z_2}) \in \text{poss}(W, \omega, d_1, i)} \{f(\mathbf{z_1}) + f(\mathbf{z_2})\} & \text{if poss}(W, \omega, d_1, i) \neq \emptyset \\ \infty & \text{otherwise,} \end{cases}$$

and

$$f_2(W, \omega, d_1, i) = 1 + \min_{d_2 \in C} \{f_1(W, \omega, d_2, i)\}.$$

Finally, for all $i > 1$, we set

$$f(W, \omega, d_1, i) = \min\{f_1(W, \omega, d_1, i), f_2(W, \omega, d_1, i), f(W, \omega, d_1, i-1)\}. \tag{4.4}$$

Observe that, for $i > 1$, the value of $f(W, \omega, d_1, i)$ depends only on values of $f(W', \omega, d', i')$ where i' is strictly smaller than i, and so a recursion based on these relationships will terminate. In the computation, we begin by initialising all the values for $i = 1$, and then given all values of f with index $i' < i$ we can compute all values of f_1 and f_2 with index i, and combine these to find all values of f with index i. We complete the table by repeating this process for all values of i from 2 to n.

To show that f has the required properties, we first prove by induction on i that we have $f(W, \omega, d_1, i) \leq \min_{T \in \mathcal{T}_i(W,G)} m_T(T, \omega, d_1)$ for each choice of W and d_1, if $\mathcal{T}_i(W, G) \neq \emptyset$. Later we will also prove the reverse inequality. We have already seen that equality holds in the base case, for $i = 1$, so let us consider the case for $i > 1$ and assume that the result holds for smaller values.

If $|W| = 1$, it is clear that

$$\min_{T \in \mathcal{T}_i(W,G)} m_T(T, \omega, d_1) = \min_{T \in \mathcal{T}_1(W,G)} m_T(T, \omega, d_1) = f(W, \omega, d_1, 1).$$

Thus we have

$$\min_{T\in\mathcal{T}_i(W,G)} m_T(T,\omega,d_1) = f(W,\omega,d_1,1) \geq f(W,\omega,d_1,2) \geq \cdots \geq f(W,\omega,d_1,i),$$

as required. So we may assume $|W| \geq 2$.

We may assume that there exists at least one subtree of G of order at most i that contains the vertices of W. Fix $T \in \mathcal{T}_i(W,G)$ such that $m_T(T,\omega,d_1) = \min_{T'\in\mathcal{T}_i(W,G)} m_{T'}(T',\omega,d_1)$ and $|T|$ is minimal. As $|T| \geq |W| \geq 2$ and ω is a proper colouring of G, T is not monochromatic under ω. Let S be an optimal sequence to flood T with colour d_1. We proceed by case analysis on the final move, α, of S.

If T is not monochromatic immediately before α, this final move must link $r \geq 2$ monochromatic components, with vertex-sets $A_1,\ldots,A_r$. We may assume α is played in A_1, and so each of $A_2,\ldots,A_r$ is adjacent only to A_1 in T. By minimality of T, we must have $1 \leq |W \cap A_l| < |W|$ for each $l \in \{2,\ldots,r\}$, so in particular $\emptyset \neq W \cap A_2 \neq W$.

Let x_1x_2 be the unique edge of T with $x_1 \in A_1$ and $x_2 \in A_2$. For $l \in \{1,2\}$, set T_l to be the component of $T - \{x_1x_2\}$ that contains x_l, $W_l = W \cap V(T_l)$, and $j_l = |T_l|$. Note that $((W_1 \cup \{x_1\},\omega,d_1,j_1),(W_2 \cup \{x_2\},\omega,d_1,j_2)) \in \mathrm{poss}(W,\omega,d_1,i)$, and also that for $l \in \{1,2\}$, $T_l \in T_{j_l}(W_l \cup \{x_l\},G)$. Set S_l to be the subsequence of S consisting of moves played in T_l, and observe that S_1 and S_2 partition S. Moreover S_l, played in T_l, makes this tree monochromatic with colour d_1, so we have $m_{T_l}(T_l,\omega,d_1) \leq |S_l|$.

Observe also that, as $j_1, j_2 < i$, the inductive hypothesis implies that, for $l \in \{1,2\}$,

$$\begin{aligned} f(W_l \cup \{x_l\},\omega,d_1,j_l) &\leq \min_{T'\in\mathcal{T}_{j_l}(W_l\cup\{x_l\},G)} m_{T'}(T',\omega,d_1) \\ &\leq m_{T_l}(T_l,\omega,d_1). \end{aligned}$$

Hence we see that

$$\begin{aligned}
f(W,\omega,d_1,i) &\leq f_1(W,\omega,d_1,i) \\
&\leq f(W_1 \cup \{x_1\},\omega,d_1,j_1) + f(W_2 \cup \{x_2\},\omega,d_1,j_2) \\
&\qquad\qquad \text{by definition} \\
&\leq m_{T_1}(T_1,\omega,d_1) + m_{T_2}(T_2,\omega,d_1) \\
&\leq |S_1| + |S_2| \\
&= |S| \\
&= m_T(T,\omega,d_1) \\
&= \min_{T' \in \mathcal{T}_i(W,G)} m_{T'}(T',\omega,d_1) \\
&\qquad\qquad \text{by choice of } T.
\end{aligned}$$

We now consider the case in which T is monochromatic in some colour $d_2 \in C$ before the final move of S, so α simply changes the colour to d_1. Note that T cannot be monochromatic before the penultimate move of S, otherwise we could obtain a shorter sequence to flood T with colour d_1. Set S' to be the initial segment of S with just the final move omitted. Then S' must be an optimal sequence to flood T with colour d_2, and does not make T monochromatic before the final move, so we can apply the reasoning above to see that

$$f_1(W,\omega,d_2,i) \leq |S'| = m_T(T,\omega,d_1) - 1.$$

But then

$$\begin{aligned} f(W,\omega,d_1,i) &\leq f_2(W,\omega,d_2,i) \\ &\leq 1 + f_1(W,\omega,d_2,i) \\ &\qquad\qquad \text{by definition} \\ &\leq m_T(T,\omega,d_1) \\ &= \min_{T' \in \mathcal{T}_i(W,G)} m_{T'}(T',\omega,d_1) \\ &\qquad\qquad \text{by choice of } T, \end{aligned}$$

completing the proof that $f(W,\omega,d_1,i) \leq \min_{T\in\mathcal{T}_i(W,G)} m_T(T,\omega,d_1)$, for every W and d_1.

To prove the correctness of the algorithm, it remains to show that for any $W \subset V$ (with $|W| \leq k$) and $d_1 \in C$ we also have $f(W,\omega,d_1,i) \geq \min_{T\in\mathcal{T}_i(W,G)} m_T(T,\omega,d_1)$. Once again, we proceed by induction on i, and note that the base case for $i = 1$ holds. Assume that $i > 1$ and that the result holds for smaller values.

In fact we will prove the following claim.

Claim. *For any $(W_1,W_2) \in \mathrm{part}(W)$, $d_1 \in C$, $x_1x_2 \in E$ with $x_1 \notin W_2$ and $x_2 \notin W_1$, and $j_1, j_2 > 0$ such that $j_1 + j_2 = i$, we have*

$$\begin{aligned} &\min_{T\in\mathcal{T}_i(W,G)} m_T(T,\omega,d_1) \leq \\ &\qquad f(W_1 \cup \{x_1\},\omega,d_1,j_1) + f(W_2 \cup \{x_2\},\omega,d_1,j_2). \end{aligned} \tag{4.5}$$

To see that it is sufficient to prove this claim, first observe that the claim imme-

diately implies that

$$\min_{T \in \mathcal{T}_i(W,G)} m_T(T, \omega, d_1) \leq f_1(W, \omega, d_1, i). \tag{4.6}$$

Moreover, it is clear that, for any $d_2 \in C$,

$$\min_{T \in \mathcal{T}_i(W,G)} m_T(T, \omega, d_1) \leq 1 + \min_{T \in \mathcal{T}_i(W,G)} m_T(T, \omega, d_2)$$

and so, if the claim holds, it follows from (4.6) that

$$\min_{T \in \mathcal{T}_i(W,G)} m_T(T, \omega, d_1) \leq 1 + f_1(W, \omega, d_2, i)$$

for any $d_2 \in C$, and hence

$$\min_{T \in \mathcal{T}_i(W,G)} m_T(T, \omega, d_1) \leq f_2(W, \omega, d_1, i).$$

Finally, note that

$$\begin{aligned} \min_{T \in \mathcal{T}_i(W,G)} m_T(T, \omega, d_1) &\leq \min_{T \in \mathcal{T}_{i-1}(W,G)} m_T(T, \omega, d_1) \\ &\qquad \text{as } \mathcal{T}_{i-1}(W,G) \subseteq \mathcal{T}_i(W,G) \\ &\leq f(W, \omega, d_1, i-1) \\ &\qquad \text{by inductive hypothesis.} \end{aligned}$$

Hence, if the claim holds, it follows that $\min_{T \in \mathcal{T}_i(W,G)} m_T(T, \omega, d_1)$ is less than or equal to every expression on the right hand side of (4.4), giving $f(W, \omega, d_1, i) \geq \min_{T \in \mathcal{T}_i(W,G)} m_T(T, \omega, d_1)$, as required. Thus it is sufficient to prove the claim.

We now prove the claim. Suppose $(W_1, W_2) \in \text{part}(W)$, $d_1 \in C$, $x_1x_2 \in E$ with $x_1 \notin W_2$ and $x_2 \notin W_1$, and $j_1, j_2 > 0$ such that $j_1 + j_2 = i$. For $l \in \{1, 2\}$, pick

$T_l \in \mathcal{T}_{j_l}(W_l \cup \{x_l\}, G)$ such that

$$m_{T_l}(T_l, \omega, d_1) = \min_{T' \in \mathcal{T}_{j_l}(W_l \cup \{x_l\}, G)} m_{T'}(T', \omega, d_1).$$

Note that, by the inductive hypothesis (as $j_l < i$), we then have

$$m_{T_l}(T_l, \omega, d_1) \leq f(W_l \cup \{x_l\}, \omega, d_1, j_l).$$

Now set $H = T_1 \cup T_2 \cup \{x_1x_2\}$, and fix a d_1-minimal spanning tree T' of H (so, by Theorem 3.1, $m_H(H, \omega, d_1) = m_{T'}(T', \omega, d_1)$). Note that $T' \in \mathcal{T}_i(W, G)$. Thus we see that

$$\begin{aligned}
\min_{T \in \mathcal{T}_i(W,G)} m_T(T, \omega, d_1) &\leq m_{T'}(T', \omega, d_1) \\
&= m_H(H, \omega, d_1) \\
&\qquad \text{by choice of } T' \\
&\leq m_{T_1}(T_1, \omega, d_1) + m_{T_2}(T_2, \omega, d_1) \\
&\qquad \text{by Corollary 3.8} \\
&\leq f(W_1 \cup \{x_1\}, \omega, d_1, j_1) + f(W_2 \cup \{x_2\}, \omega, d_1, j_2),
\end{aligned}$$

completing the proof of the claim.

It remains only to bound the time taken to compute $f(U, \omega, d, n)$. Note that each value $f(W, \omega, d_1, 1)$ (for any $W \subset V$ of size at most k and $d_1 \in C$) can be computed in constant time.

Suppose we have computed the value of $f(W, \omega, d_1, i)$ for each $d_1 \in C$ and each $W \subset V$ of size at most k. To compute $f_1(W', \omega, d_2, i+1)$ for any W' and d_2, we take the minimum over at most 2^k ways to partition a set of up to k points, the $|E|$ edges in the graph, the $|C|$ colours in the initial colouring, and the $2(i-1)$ ordered

pairs of positive integers that sum to i. Thus we take the minimum over a set of $O(2^k \cdot |E| \cdot |C| \cdot n)$ values, each of which can be computed in time $O(n)$ by adding a pair of existing values in the table, and so compute $f_1(W', \omega, d_2, i+1)$ in time $O(2^k \cdot |E| \cdot |C| \cdot i \cdot n) = O(2^k \cdot |E| \cdot |C| \cdot n^2)$.

Once we have computed the value of f_1 for all entries with index $i+1$, we can compute f_2 for each such entry in time $O(|C|)$. Given the values of f_1 and f_2 for each entry with index $i+1$, and the values of f for entries with index i, we can compute f for any entry with index $i+1$ in constant time.

Thus in total we require time at most $O(2^k \cdot |E| \cdot |C| \cdot n^2)$ to compute the value of f for each entry in the table. In total, the table contains $O(n^{k+1} \cdot |C|)$ entries (as there are $O(n^k)$ subsets of size at most k, a choice of $|C|$ colours, and i takes integer values in the range $[1, n]$), so we can compute all entries, and hence determine $f(U, \omega, d, n)$, in time $O(n^{k+3} \cdot |E| \cdot |C|^2 \cdot 2^k)$. □

Chapter 5

Flood-It Played on Rectangular Boards of Fixed Height

It is known that Fixed-Flood-It and Free-Flood-It are NP-hard on traditional $n \times n$ boards (if at least three colours are present initially) [8], and on the other hand that Free-Flood-It can be solved in polynomial time on $1 \times n$ boards (this follows immediately from Corollary 4.7); thus, a natural question is to investigate the complexity of the problems when restricted to rectangular $k \times n$ boards, for various fixed values of k.

In [16], Clifford, Jalsenius, Montanaro and Sach prove that Fixed-Flood-It is solvable in time $O(n)$ on $2 \times n$ boards, whereas both Fixed-Flood-It and Free-Flood-It are NP-hard even when restricted to $3 \times n$ boards (when there is no bound on the number of colours used). We improve on this hardness result in Section 5.1, showing that the problems remain hard on $3 \times n$ boards even if only four colours are used. In Section 5.2 we show another hardness result, demonstrating that Free-Flood-It remains NP-hard when restricted to $2 \times n$ boards; we consider the parameterised complexity of c-Free Flood It on such boards (with parameter c) in the next chapter. The hardness of Free-Flood-It on $2 \times n$ boards is somewhat

	$c = 2$	$c = 3$	$c = 4$	c unbounded
$1 \times n$	P (Theorem 4.2)	P (Corollary 4.7)	P (Corollary 4.7)	P (Corollary 4.7)
$2 \times n$	P (Theorem 4.2)	See Table 5.2		
$3 \times n$	P (Theorem 4.2)	?	NP-h (Section 5.1)	NP-h [16]
$n \times n$	P (Theorem 4.2)	NP-h [8]	NP-h [8]	NP-h [8]

Table 5.1: The complexity status of c-Fixed-Flood-It and c-Free-Flood-It on rectangular boards of different heights, when c colours are present on the initial board. (NP-h = NP-hard, P = polynomial-time, ? = unknown.)

	c constant	c unbounded
c-Fixed-Flood-It	P [16]	P [16]
c-Free-Flood-It	P (Chapter 6)	NP-h (Section 5.2)

Table 5.2: The complexity status of flood-filling problems on $2xn$ boards.

surprising, as this gives the first class of graphs for which the complexities of the fixed and free variants are known to differ. The complexity status of the various flood-filling problems restricted to rectangular boards of different heights is summarised in Tables 5.1 and 5.2.

When we consider the game played on a rectangular board B, we are effectively playing the game in a corresponding coloured graph G_B, obtained from the planar dual of B (in which there is one vertex corresponding to each square of B, and vertices are adjacent if they correspond to squares which are either horizontally or vertically adjacent in B) by giving each vertex the colour of the corresponding square in B. We identify areas of B with the corresponding subgraphs of G_B, and may refer to them interchangeably.

Before going on to consider the complexity of the problems on $3 \times n$ and $2 \times n$ boards, we observe that Theorem 4.6 can be applied to give an additive approximation to c-Free-Flood-It on $k \times n$ boards, for any fixed c and k.

Corollary 5.1. *For any fixed k, we can compute a constant additive approximation to c-Free-Flood-It, restricted to $k \times n$ boards, in time $O(n^4)$.*

Proof. Let G, with colouring ω (using at most c colours) be the coloured graph

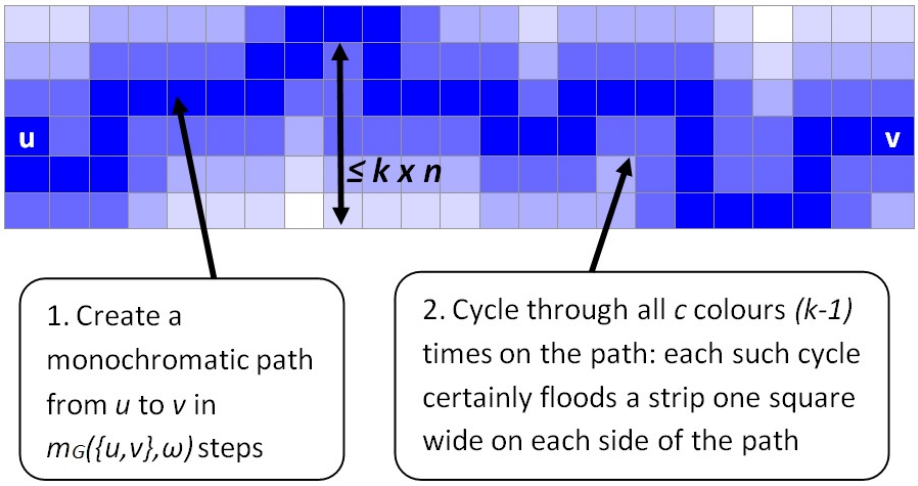


Figure 5.1: A strategy to flood a $k \times n$ board.

corresponding to a $k \times n$ Flood-It board, and let u (respectively v) be a vertex corresponding to a square incident with the left-hand (respectively right-hand) edge of the board. Suppose the sequence of moves S floods G. Then S clearly links u and v, so we have $|S| \geq m_G(\{u, v\}, \omega)$. But observe also that one strategy to flood the board would be to create a monochromatic path from u to v, and then cycle through all c colours present on the board at most $k-1$ times, to absorb all remaining regions, as illustrated in Figure 5.1. Thus we have $m_G(G, \omega) \leq m_G(\{u, v\}, \omega) + c(k-1)$. Hence

$$m_G(\{u, v\}, \omega) \leq m_G(G, \omega) \leq m_G(\{u, v\}, \omega) + c(k-1),$$

and $m_G(\{u, v\}, \omega)$ gives an additive approximation to $m_G(G, \omega)$.

By Theorem 4.6, we can compute $m_G(\{u, v\}, \omega)$ in time $O(n^4)$ for fixed k and c, thus obtaining in polynomial time an additive approximation to $m_G(G, \omega)$. □

5.1 Hardness of Fixed-Flood-It and Free-Flood-It on $3 \times n$ boards

In contrast to our approximation result in Corollary 5.1, we see in this section that, even for small values of k, it remains NP-hard to solve flood-filling problems *exactly* on $k \times n$ boards.

In particular, we show that both 4-Fixed-Flood-It and 4-Free-Flood-It are NP-hard on $3 \times n$ boards. This improves on the result of Clifford et. al. [16] that Fixed-Flood-It and Free-Flood-It are NP-hard on such boards when the number of colours is unbounded.

Both our results are proved by means of reductions from the decision version of Shortest Common Supersequence (SCS), shown to be NP-complete over a binary alphabet by Räihä and Ukkonen in [48]. This is the same problem used in [8] to prove the NP-hardness of flood-filling problems on $n \times n$ boards.

Shortest Common Supersequence (SCS)
Input: Strings $s_1, \ldots, s_k$ (of length at most w) over a binary alphabet $\Sigma = \{1, 2\}$, and an integer l.
Question: Do $s_1, \ldots, s_k$ have a common supersequence of length at most l?

We will construct $3 \times n$ boards B and B' for the 4-FIXED-FLOOD-IT and 4-FREE-FLOOD-IT problems respectively (each using colours $\{1, 2, 3, 4\}$), so that these boards can be flooded with at most $2l+3$ moves if and only if $s_1, \ldots s_k$ have a common supersequence of length at most l.

5.1.1 The 4-Fixed-Flood-It case

We prove the following theorem.

Theorem 5.2. *4-Fixed-Flood-It is NP-hard when restricted to* $3 \times n$ *boards.*

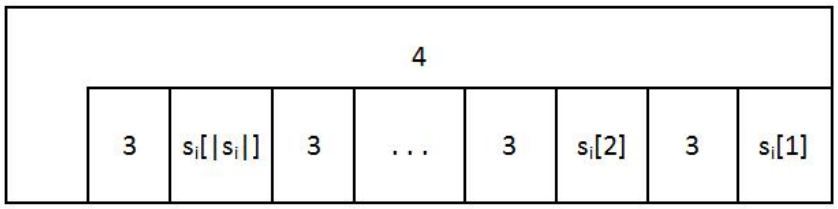


Figure 5.2: The gadget G_i

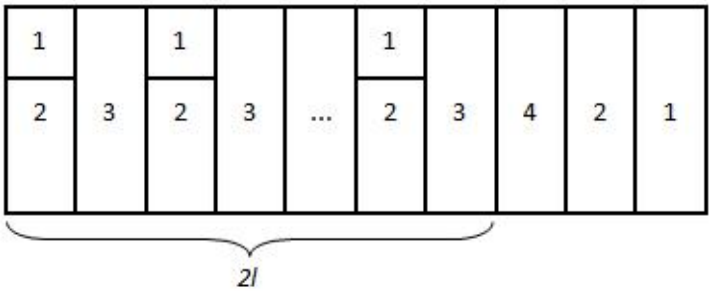


Figure 5.3: The rectangle R

To show the reduction, we construct a $3 \times n$ Flood-It board with four colours as follows. For each s_i, we include a $2 \times (2|s_i| + 1)$ gadget G_i as illustrated in Figure 5.2, where $s_i[j]$ denotes the j^{th} character of s_i. The idea of alternating colours that correspond to the symbols in the binary alphabet with a third "dummy" colour (to separate components corresponding to repeated symbols) was used in [8] for the proof of hardness on $n \times n$ boards; to prove hardness on $3 \times n$ boards we need an additional colour, to ensure we can still control the order in which the parts of the gadgets encoding the sequences must be flooded.

We then place these in a $3 \times n$ board filled with colour 3 as illustrated in Figure 5.4, and add a section R at the end, where R is as shown in Figure 5.3. (R is similar to a gadget used in the $n \times n$ construction from [8], and has the same role of ensuring that optimal sequences of moves must have a specific structure.) Note we can take $n \leq k(2w + 2) + 2l + 3$.

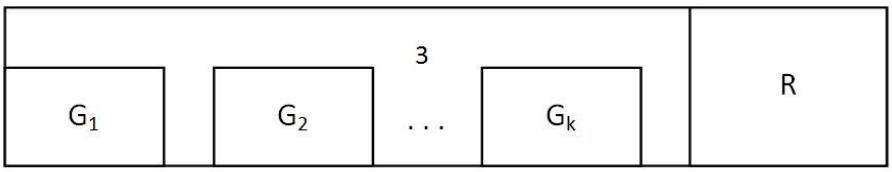


Figure 5.4: The board B

We now show, in the next two lemmas, that $s_1, \ldots, s_k$ have a common supersequence of length at most l if and only if we can flood B in $2l + 3$ steps.

Lemma 5.3. *If $s_1, \ldots, s_k$ have a common supersequence of length at most l, then we can flood the board B (starting from the top left) in $2l + 3$ steps.*

Proof. Let $a_1 \ldots a_l$ be a common supersequence of length exactly l (padding a shortest sequence with 1s if necessary). Then we claim that the sequence of moves $a_1 3 a_2 3 a_3 3 \ldots a_{l-1} 3 a_l 3421$, of length $2l + 3$, floods the board. First observe that this sequence floods R: each move extends the external monochromatic area into R by at least part of one column, and the final two moves of colours 2 and 1 respectively will flood any remaining unflooded partial columns. But this sequence will also flood G_i for each i: s_i is a subsequence of $a_1 \ldots a_l$ so $s_i[1]3s_i[2] \ldots s_i[|s_i|]3$ is a subsequence of $a_1 3 a_2 3 \ldots a_l 3$, and the first $2l$ moves will flood all of G_i not coloured 4, before the $(2l + 1)^{st}$ move floods the region coloured 4. So this sequence of $2l + 3$ moves does indeed flood B. □

Lemma 5.4. *If we can flood B in at most $2l + 3$ steps, starting from the top left corner at each move, then $s_1, \ldots s_k$ have a common supersequence of length at most l.*

Proof. First observe that we cannot flood R from the outside in fewer than $2l + 3$ steps, as each move can only extend the external monochromatic area to the right by one column. Moreover, any sequence of $2l + 3$ moves that floods R must consist of l 1s or 2s, alternated with 3s, then finally $4, 2, 1$.

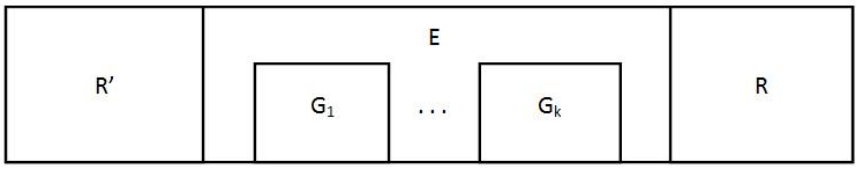


Figure 5.5: The board B'

Suppose such a sequence $c_1, \ldots c_{2l+3}$ also floods every G_i. Note that the external area never has colour 3 after the only move of colour 4. So, in order to flood the leftmost square of colour 3 in each G_i, we must in fact flood the bottom row of each G_i sequentially from the right, and moreover we must have flooded this row by the end of the $(2l)^{th}$ move. But then, for each i, s_i must be a subsequence of $c_1, \ldots c_{2l}$ restricted to $\{1, 2\}$, which is a sequence of length l. Hence we have a common supersequence of $s_1, \ldots s_k$ of length l, as required. □

Proof of Theorem 5.2. The reduction from Shortest Common Supersequence follows immediately from Lemmas 5.3 and 5.4. □

5.1.2 The 4-Free-Flood-It case

We now prove an analogous theorem for the free variant of the game.

Theorem 5.5. *4-Free-Flood-It remains NP-hard when restricted to* $3 \times n$ *boards.*

The construction of the Flood-It board B' used to prove Theorem 5.5 is very similar to that in the previous section. The only difference is that we also include a second rectangular section, R', located at the left-hand end of the board (as illustrated in Figure 5.5). R' is identical to R except that it is reflected in a vertical axis. In this case we can take $n \leq k(2w + 2) + 4l + 7$.

It remains to show that $s_1, \ldots, s_k$ have a common supersequence of length at most l if and only if we can flood B' in $2l + 3$ steps.

Lemma 5.6. *If $s_1, \ldots, s_k$ have a common supersequence of length at most l, then we can flood the board B' in $2l + 3$ steps.*

Proof. We can use exactly the same strategy as described in Lemma 5.3, playing in the external area E at each move. □

Lemma 5.7. *If we can flood B' in at most $2l + 3$ moves, making moves anywhere on the board, then $s_1, \ldots, s_k$ have a common supersequence of length at most l.*

Proof. First observe that we need at least $2l + 3$ moves to flood the board: initially the minimum number of monochromatic areas on any end-to-end path is $4l + 7$, and as each move can decrease this by at most two, we do indeed require a minimum of $2l + 3$ moves. Moreover, to achieve this lower bound, every move must reduce the number of monochromatic areas lying on an end-to-end path by exactly two. One consequence of this is that no move can be played inside any G_i.

Another consequence is that we can only make a move of colour 4 once in any optimal sequence: only two regions of this colour lie on any shortest end-to-end path, and so we can only make one colour 4 move that will decrease the path length as required. However, to flood all the G_i, this single move of colour 4 must be played in E, so we cannot play colour 4 until the area A, containing E and adjacent to both colour 4 regions in $R \cup R'$, has been linked. It requires at least $2l$ moves to link A, so we make at least $2l + 1$ moves up to and including the move of colour 4. These moves have no effect on the regions of colour 1 and 2 at the ends of the board, and it requires at least two moves to flood these remaining end-regions, so we can only possibly flood B in $2l+3$ moves if we link A (except for regions of colour 4) in exactly $2l$ moves, and then play colours 4,2, and 1 in the external area.

As colour 3 is then never played after colour 4 we see, as before, that the left-most square of colour 3 in each G_i can only be flooded if the bottom row of each G_i is flooded sequentially from the right, and this must be done within the first $2l$ moves.

Thus, if s is the subsequence of the first $2l$ moves consisting of those that are made in an area containing E and are of colour 1 or 2, then s is a common supersequence of $s_1, \ldots, s_k$.

To complete the proof it therefore suffices to show that $|s| \leq l$. But every move in s reduces by two the number of monochromatic areas lying on the shortest end-to-end path, by means of flooding two areas of colour $c \in \{1, 2\}$. Initially there were only $2l + 4$ regions of colour 1 or 2 on any shortest end-to-end path, and four of these we know are not flooded within the first $2l$ moves, so $|s|$ can be at most l, as required. □

Proof of Theorem 5.5. The reduction from Shortest Common Supersequence follows immediately from Lemmas 5.6 and 5.7. □

5.2 Hardness of Free-Flood-It on $2 \times n$ boards

In this section we prove the following theorem.

Theorem 5.8. Free-Flood-It *remains NP-hard when restricted to* $2 \times n$ *boards.*

This is somewhat surprising, as we will see in the following chapter that c-Free-Flood-It can be solved in polynomial time on $2 \times n$ boards for fixed c, while [16] gives a linear time algorithm to solve Fixed-Flood-It in this situation. We demonstrate here that the problem is almost certainly not in **P** if we remove both these restrictions (that moves are always played at the same vertex, or the number or colours is bounded). This is the first class of graphs for which such a result has been shown.

The proof is by means of a reduction from Vertex Cover, shown to be NP-hard by Karp in [37].

Vertex Cover
Input: A graph $G = (V, E)$, and an integer k.
Question: Does there exist $U \subseteq V$ with $|U| \leq k$ such that every edge in E has

at least one endpoint in U?

Before giving the details of our reduction in Section 5.2.2, we prove some general results which will be needed to demonstrate its correctness.

5.2.1 Auxiliary results

We begin by showing that there exists an optimal sequence to flood any tree in which no moves are played at leaf vertices. Given any tree T, we denote by $\text{trunk}(T)$ the subtree obtained by deleting all leaves of T.

Lemma 5.9. *Let T be any tree, and ω a colouring of the vertices of T. Then there exists a sequence of moves S, of length $m_T(T, \omega)$, which makes T monochromatic and in which all moves are played in* $\text{trunk}(T)$.

Proof. Let S_0 be any optimal sequence to flood T, and set S_0' to be the subsequence of S_0 consisting of moves that change the colour of a vertex in $\text{trunk}(T)$. Note that we may assume without loss of generality that all moves of S_0' are played in $\text{trunk}(T)$. Note further that $S_0 \setminus S_0'$ contains only moves played at leaves, and let U be the set of leaves in which moves of $S_0 \setminus S_0'$ are played. Observe that playing S_0' in T will make $T \setminus U$ monochromatic, and so we can flood the entire tree by playing a sequence S which consists of S_0' followed by a further $|\operatorname{col}(U, \omega)|$ moves, cycling through the colours still present in leaves of T (playing all moves in $\text{trunk}(T)$). Thus

$$|S_0| \leq |S_0'| + |\operatorname{col}(U, \omega)| \leq |S_0'| + |U|.$$

However, it is clear that $|S_0| \geq |S_0'| + |U|$, as $S_0 \setminus S_0'$ contains at least one move played at each vertex in U. Hence we see that $|S| \leq |S_0|$, and so S is an optimal sequence to flood T in which all moves are played in $\text{trunk}(T)$, as required. □

We now give a relationship between the number of moves required to flood the

same graph with different initial colourings.

Lemma 5.10. *Let G be a connected graph, and let ω and ω' be two colourings of the vertices of G (from colour-set C). Let $\mathcal{A}$ be the set of all maximal monochromatic components of G with respect to ω', and for each $A \in \mathcal{A}$ let c_A be the colour of A under ω'. Then, for any $d \in C$,*

$$m_G(G, \omega, d) \leq m_G(G, \omega', d) + \sum_{A \in \mathcal{A}} m_A(A, \omega, c_A).$$

Proof. We proceed by induction on $m_G(G, \omega', d)$. Note that if $m_G(G, \omega', d) = 0$ then the result is trivially true: in this case $\mathcal{A}$ contains a single monochromatic component G, with colour d, so we have

$$m_G(G, \omega', d) + \sum_{A \in \mathcal{A}} m_A(A, \omega, c_A) = 0 + \sum_{A \in \{G\}} m_A(A, \omega, c_A) = m_G(G, \omega, d).$$

Suppose now that $m_G(G, \omega', d) > 0$, and let S be an optimal sequence of moves to flood G with colour d, when the initial colouring is ω'. We proceed by case analysis on the final move, α, of S. First suppose that G is already monochromatic before α, so this final move just changes the colour of the entire graph to d from some colour $d' \in C$. In this case, $m_G(G, \omega', d) = m_G(G, \omega', d') + 1$, and so we may apply the inductive hypothesis to see that

$$\begin{aligned} m_G(G, \omega, d) &\leq 1 + m_G(G, \omega, d') \\ &\leq 1 + m_G(G, \omega', d') + \sum_{A \in \mathcal{A}} m_A(A, \omega, c_A) \\ &= m_G(G, \omega', d) + \sum_{A \in \mathcal{A}} m_A(A, \omega, c_A), \end{aligned}$$

as required.

Now suppose that G is not monochromatic before α, and so this move links monochromatic components $X_1, \ldots, X_r$. We may assume that α changes the colour of X_1 from d' to d, and that all the components $X_2, \ldots, X_r$ have colour d before α. Let S_i denote the subsequence of S consisting of moves played in X_i, and observe that playing S_i in the isolated subgraph X_i must flood this graph with colour d, so $m_{X_i}(X_i, \omega', d) \leq |S_i|$. Note that, as no move can split a monochromatic component, the sets $\mathcal{A}_i = \{A \in \mathcal{A} : A \subseteq X_i\}$ (for $1 \leq i \leq r$) partition $\mathcal{A}$.

Observe that, for $2 \leq i \leq r$, $m_{X_i}(X_i, \omega, d) < |S| = m_G(G, \omega', d)$, and so we may apply the inductive hypothesis to see that

$$\begin{aligned} m_{X_i}(X_i, \omega, d) &\leq m_{X_i}(X_i, \omega', d) + \sum_{A \in \mathcal{A}_i} m_A(A, \omega, c_A) \\ &\leq |S_i| + \sum_{A \in \mathcal{A}_i} m_A(A, \omega, c_A). \end{aligned}$$

Similarly, the inductive hypothesis gives

$$m_{X_1}(X_1, \omega, d') \leq m_{X_1}(X_1, \omega', d') + \sum_{A \in \mathcal{A}_1} m_A(A, \omega, c_A),$$

and so, as $m_{X_1}(X_1, \omega', d') \leq |S_1| - 1$, we have

$$\begin{aligned} m_{X_1}(X_1, \omega, d) &\leq 1 + m_{X_1}(X_1, \omega, d') \\ &\leq |S_1| + \sum_{A \in \mathcal{A}_1} m_A(A, \omega, c_A). \end{aligned}$$

Now we can apply Corollary 3.8 to see that

$$m_G(G, \omega, d) \leq \sum_{i=1}^{r} m_{X_i}(X_i, \omega, d),$$

and so

$$\begin{aligned} m_G(G, \omega, d) &\leq \sum_{i=1}^{r} (|S_i| + \sum_{A \in \mathcal{A}_i} m_A(A, \omega, c_A)) \\ &= |S| + \sum_{A \in \mathcal{A}} m_A(A, \omega, c_A) \\ &= m_G(G, \omega', d) + \sum_{A \in \mathcal{A}} m_A(A, \omega, c_A), \end{aligned}$$

completing the proof. □

Finally, we show that there is an optimal sequence to flood any tree in which the total number of maximal monochromatic components decreases at each stage.

Lemma 5.11. *Let T be any tree, and ω a colouring of the vertices of T. Then there exists a sequence of moves S, of length $m_T(T, \omega, d)$, which makes T monochromatic and in which every move, except possibly the last, strictly decreases the number of maximal monochromatic components of T.*

Proof. We proceed by induction on $m_T(T, \omega, d)$. Note that the result is trivially true in the base case, for $m_T(T, \omega, d) = 0$. Assume, therefore, that $m_T(T, \omega, d) > 0$ and let S_0 be any optimal sequence to flood T with colour d, and that the result holds for any T' with $m_{T'}(T', \omega, d) < m_T(T, \omega, d)$. Suppose the first move of S_0 is α.

First suppose that α strictly decreases the number of maximal monochromatic components of T. Then we can apply the inductive hypothesis to see that there exists a sequence of moves S' that will take $m_T(T, \alpha(\omega, T), d)$ steps to flood T with colour d, starting from the colouring $\alpha(\omega, T)$, and such that every move of S' strictly decreases the number of monochromatic components of T. Thus, if we set S to be the sequence of moves in which we first play α and then play S', this sequence S will have the required properties.

Now suppose that this is not the case, so $\alpha = (v, d')$ for some $v \in V(T)$ and $d' \in C$ such that no neighbours of v in T have colour d' under ω. Note that we may assume

without loss of generality that ω is a proper colouring of T, so α only changes the colour of v. If in fact $V(T) = \{v\}$, then any optimal sequence to flood T will have at most one move, so the result is trivially true. Otherwise, there must be some move which links v to an adjacent monochromatic component. Set β to be the first such move, and let $\bar{S}_0$ be the initial segment of S_0 up to and including β.

Suppose that β links v to a neighbour u. We then define $\alpha' = (v, \mathrm{col}(u, \omega))$, and claim that there exists an optimal sequence of moves to flood T with colour d that has α' as the first move; with such a sequence, we would be in the previous case and so the result would follow by the inductive hypothesis. Thus it suffices to prove this claim, that is to show that $m_T(T, \alpha'(\omega, T), d) \leq m_T(T, \omega, d) - 1$.

Let $A_1, \ldots, A_r$ be the monochromatic components of T with respect to the colouring $\bar{S}_0(\omega, T)$, where A_i has colour d_i under this colouring, and assume without loss of generality that $v \in A_1$. For $1 \leq i \leq r$, set S_i to be the subsequence of $\bar{S}_0$ consisting of moves played in A_i; note that these subsequences partition $\bar{S}_0$ and that, for each i, we have $m_{A_i}(A_i, \omega, d_i) \leq |S_i|$.

By Lemma 5.10 we see that

$$\begin{aligned} m_T(T, \alpha'(\omega, T), d) &\leq m_T(T, \bar{S}_0(\omega, T), d) + \sum_{i=1}^{r} m_{A_i}(A_i, \alpha'(\omega, T), d_i) \\ &\leq |S_0| - |\bar{S}_0| + \sum_{i=1}^{r} m_{A_i}(A_i, \alpha'(\omega, T), d_i) \\ &= m_T(T, \omega, d) - |\bar{S}_0| + \sum_{i=1}^{r} m_{A_i}(A_i, \alpha'(\omega, T), d_i), \end{aligned}$$

and so it suffices to prove that

$$\sum_{i=1}^{r} m_{A_i}(A_i, \alpha'(\omega, T), d_i) \leq |\bar{S}_0| - 1. \tag{5.1}$$

First observe that, for $2 \leq i \leq r$, we have $\alpha'(\omega, T)|_{A_i} = \omega|_{A_i}$, and so

$$m_{A_i}(A_i, \alpha'(\omega, T), d_i) = m_{A_i}(A_i, \omega, d_i) \leq |S_i|.$$

Now consider A_1, and note that A_1 is not monochromatic before β; suppose that, immediately before β, A_1 has maximal monochromatic components $A_1^1, \ldots, A_1^l$. Recall that v is not linked to any other vertex before β, so without loss of generality suppose that $A_1^1 = \{v\}$; we may further assume that $u \in A_1^2$.

We now set S_1^j to be the subsequence of S_1 consisting of moves played in A_1^j; note that these subsequences partition S_1, and that for each i, $m_{A_1^j}(A_1^j, \omega, d_1) \leq |S_1^j|$. Note further that, for $j \geq 2$, we have $\alpha'(\omega, T)|_{A_1^j} = \omega|_{A_1^j}$, and so

$$m_{A_1^j}(A_1^j, \alpha'(\omega, T), d_1) = m_{A_1^j}(A_1^j, \omega, d_1) \leq |S_1^j|.$$

Observe also that $T[V(A_1^1 \cup A_1^2)]$ with colouring $\alpha'(\omega, T)$ is identical, after contracting monochromatic components, to A_1^2 with colouring ω, and so we see that

$$m_{T[V(A_1^1 \cup A_1^2)]}(T[V(A_1^1 \cup A_1^2)], \alpha'(\omega, T), d_1) = m_{A_1^2}(A_1^2, \omega, d_1).$$

Then, by Corollary 3.8, we see that

$$\begin{aligned} m_{A_1}(A_1, \alpha'(\omega, T), d_1) &\leq m_{T[V(A_1^1 \cup A_1^2)]}(T[V(A_1^1 \cup A_1^2)], \alpha'(\omega, T), d_1) \\ &\qquad + \sum_{j=3}^{l} m_{A_1^j}(A_1^j, \alpha'(\omega, T), d_1) \\ &\leq \sum_{j=2}^{l} m_{A_1^j}(A_1^j, \omega, d_1) \\ &\leq \sum_{j=2}^{l} |S_1^j| \\ &= |S_1| - |S_1^1|. \end{aligned}$$

Note that $\alpha \in S_1^1$, so $|S_1^1| \geq 1$, implying that in fact

$$m_{A_1}(A_1, \alpha'(\omega, T), d) \leq |S_1| - 1.$$

But then

$$\sum_{i=1}^{r} m_{A_i}(A_i, \alpha'(\omega, T), d_i) \leq (\sum_{i=1}^{r} |S_i|) - 1 = |\bar{S_0}| - 1,$$

and so (5.1) holds, which completes the proof. □

5.2.2 The reduction

Given a graph $G = (V, E)$, we construct a $2 \times n$ Flood-It board B_G as follows. Suppose $E = \{e_1, \ldots, e_m\}$. For each edge $e = uv \in E$ we construct the gadget G'_e, as illustrated in Figure 5.6. We will refer to the single-square components incident with the bottom edge in G'_e as *islands*. G'_e is then embedded in the larger gadget G_e, as shown in Figure 5.7. Distinct colours $x_1^e, \ldots, x_r^e$ are used for each e, where $r = 2m + |V| + 2$. We then obtain the board B_G by placing these gadgets G_e in a row, as illustrated in Figure 5.8. Observe that we can take $n = m(2r + 6) = 2m(2m + |V| + 5)$. Let us also set $N = mr + 2m - 1$.

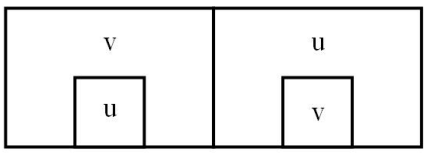


Figure 5.6: The gadget G'_e

In the following two lemmas, we prove that this board B can be flooded in $N + k$ steps if and only if G has a vertex cover of size at most k.

Lemma 5.12. *If G has a vertex cover of size at most k, then we can flood the board B_G in $N + k$ steps.*

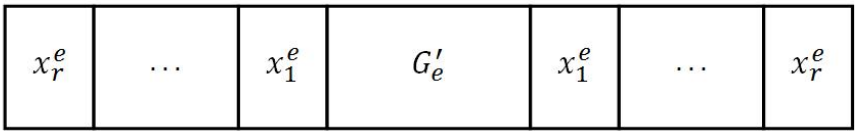


Figure 5.7: The gadget G_e

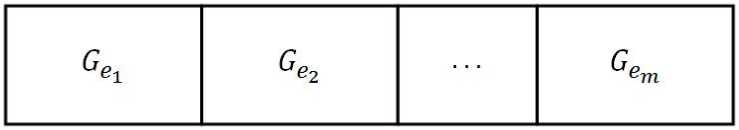


Figure 5.8: The board B

Proof. First observe that, if $e = uv$, then with $(r+1)$ moves we can flood the gadget G_e, except for a single island of colour $c(e) \in \{u, v\}$, so that it is monochromatic in colour x_r^e: first play a single move to make all of G'_e except for a single island monochromatic, then play colours $x_1^e, \ldots, x_r^e$ in this central component. Ignoring the islands for the moment, the components corresponding to each G_e now have distinct colours, so we can link these components with a minimum of $m-1$ moves. Finally, we need to flood the islands, and this requires exactly $|\{c(e) : e \in E\}|$ moves. But we know that G has a vertex cover of size at most k, say V'. By the definition of a vertex cover, if the gadget G'_e uses colours u and v, then at least one of $u, v \in V'$. So for each G_e, we may choose to leave an island of colour d where $d \in V'$. Following this strategy, we are left in the final stage with islands of at most k distinct colours, and can flood these in k steps (by cycling through each colour in turn in the external monochromatic component). Hence we can flood B_G in $N + k$ steps. □

Lemma 5.13. *If we can flood B_G in $N + k$ steps (for some $0 \leq k \leq |V|$), then G has a vertex cover of size at most k.*

Proof. Suppose the sequence S floods B_G, where $|S| = N + k$. Observe that, if we contract monochromatic components of the coloured graph corresponding to B_G, we

obtain a tree T (consisting of a path with $2m$ pendant leaf vertices); we will denote by ω the colouring this tree inherits from B_G. Let P be the unique path in T joining the two vertices in T that correspond to the monochromatic components incident with opposite ends of the board and note that, by Lemma 5.9, we may assume that all moves of S are played in $\text{trunk}(T) \subseteq P$; moreover S must flood P when played in this isolated path. By Lemma 5.11 (since the proof only changes the order of the moves, not the particular vertices at which they are played) we may further assume that every move in S decreases the number of monochromatic components of T by at least one.

We will say that a component of colour d is *eliminated* by the move α if α changes the colour of that component, linking it to an adjacent component of colour $d' \neq d$. We say that α *eliminates the colour* d if it eliminates the last component of colour d remaining in the graph.

We begin by observing the first move played in each gadget G_e decreases the number of monochromatic components on P by exactly one. Since we assume that every move decreases the number of monochromatic components of T by at least one, it follows immediately that we decrease the number of monochromatic components on P by at least one unless this move links an island to the path; however, it is clear that if the first move played in any G_e links an island to the path then this will in fact also reduce the number of monochromatic components on P by exactly one. So it remains to show that the first move in each G_e cannot decrease the number of monochromatic components on P by more than one. Suppose that this first move changes the colour of the vertex z, linking it to at least one neighbour on P. It is only possible for the move that eliminates z to decrease the number of monochromatic components of P by two if both neighbours of z on P have the same colour; but this is clearly not possible if both of these neighbours lie in G_e. Thus it must be that z has a neighbour outside G_e, and so z has colour x_r^e. But then a vertex in a different gadget $G_{\bar{e}}$ would

at some point have had to have its colour changed to x^e_{r-1} by some move; the first such move could not possibly decrease the number of monochromatic components of T, contradicting our assumption that every move of S does decrease the number of monochromatic components of T.

We now consider the moves that link islands to P, and in particular set I to be the subset of islands such that $v \in G_e$ belongs to I if and only if v is the second island in G_e to be linked to P; set $U = \mathrm{col}(I, \omega)$. By definition, for every $e = uv \in E$, we have $\{u, v\} \cap U \neq \emptyset$, and so U is a vertex cover for G. In the remainder of the proof we will show that we must in fact have $|U| \leq k$.

We claim that, for every $v \in U$, there exists a move which gives a component of P colour v but does not decrease the number of monochromatic components lying on P. Note that every move that links a vertex from I of colour v to the path must give colour v to a component of P, as no moves are played in leaves. There exists at least one such move for each $v \in U$, and if for each v one of these moves does not decrease the number of monochromatic components on P then we are done; otherwise, there must be a move α which changes the colour of a component X in order to link an island from I of colour v to the path (where $v \in G'_e$ for some $e \in E$), and which also links X to some component Y of P. Note that Y must have colour v immediately before α. There two possibilities.

1. Y contains a vertex y which originally had colour v and has never had its colour changed. Since the other island in G_e has already been linked to the path (and this must have been done by changing the colour of the island's neighbour on the path, the only other vertex in G_e which initially had colour v), this vertex y must belong to $G'_{\bar{e}}$ for some $\bar{e} \neq e$. Without loss of generality, suppose that $G'_{\bar{e}}$ lies to the right of G'_e. Let Q be the segment of P containing all vertices of X and those vertices of Y that do not lie to the right of y. Suppose that the number of colours appearing in Q under the initial colouring is i, so at least $i-1$

moves of the sequence up to and including α must be played on Q. Note that these moves cannot have any effect on vertices lying to the right of y, as y has not changed colour before α is played, so (as X is a maximal monochromatic component before α) the moves played on Q up to this point do not change the colour of any vertices that do not lie on Q. The number of colours in the initial colouring of $P \setminus Q$ is at least $mr - i + r$, as there are initially at least mr colours in total, and colours $x_1^{\bar{e}}, \ldots, x_r^{\bar{e}}$ appear both on Q and on $P \setminus Q$. All but at most one of these colours must be eliminated by moves that are either played after α or are not played on Q, so in total we have

$$|S| \geq i - 1 + mr - i + r - 1 = N + |V| + 1,$$

contradicting our initial assumption that $|S| = N + k$ for some $k \leq |V|$.

2. Every vertex in Y that initially had colour v has at some point had its colour changed, so every vertex of Y must have had its colour changed to v by a move played before α. Suppose β was the first move that gave a monochromatic component Z of Y colour v, subject to the condition that no moves after β and before α change the colour of Z. If β decreased the number of monochromatic components on P, then it must have linked Z to an adjacent vertex y that either had colour v initially or was given colour v by a previous move, and we must have $y \in Y$(as Y is a maximal monochromatic component at a point after β has been played). Note that the colour of y cannot change after β and before α, as this would mean the colour of Z also changes, contradicting our choice of β. Therefore some move before β must have given y colour v, since even if y had colour v initially it must, by assumption, change colour at some point before α is played. But then there must have been a move played before β which gave a component of Y colour v, where this component's colour does not change again

after this move and before α: this contradicts our choice of β as the first such move. Hence β must give a component of P colour v but does not decrease the number of monochromatic components lying on P, and so is the move we require.

Thus we see that there are at least $|U|$ moves in S which do not decrease the number of monochromatic components on P. Then, since we know that at least one move in every G_e decreases the number of monochromatic components on P by exactly one, no move can decrease this number by more than two, and initially there are $2mr + 2m$ monochromatic components on P, we see that

$$|S| \geq mr + m + m + |U| - 1 = mr + 2m - 1 + |U| = N + |U|,$$

and hence $|U| \leq k$. So U is a vertex cover of G of size at most k, as required. □

Proof of Theorem 5.8. The reduction from Vertex Cover is immediate from Lemmas 5.12 and 5.13. □

Chapter 6

The parameterised complexity of c-Free-Flood-It on $2 \times n$ boards

We saw in the previous chapter that Free-Flood-It remains NP-hard when restricted to $2 \times n$ boards. In contrast, in this chapter, we give an algorithm to solve c-Free-Flood-It in this situation. More specifically, we prove the following result, which shows that c-Free-Flood-It, restricted to $2 \times n$ boards, is fixed parameter tractable, parameterised by c.

Theorem 6.1. *When restricted to $2 \times n$ boards, c-Free-Flood-It can be solved in time $O(n^{10} \cdot 2^c)$.*

Together with the hardness result for $2 \times n$ boards in the previous chapter, this answers an open question of Clifford et. al. [16].

We begin in Section 6.1 with some background and auxiliary results, and then describe the algorithm in Section 6.2. First we need to introduce some additional notation and definitions that are specific to $2 \times n$ boards.

Suppose B is a $2 \times n$ Flood-It board. We define a *border* of B to be a union of edges of squares on the original board B that forms a path from the top edge of the board to the bottom (but not including any edges that form the top or bottom edge

of the board). Thus, a border in B corresponds to an edge-cut in the corresponding graph. Observe that a border is uniquely defined by the points at which it meets the top and bottom of the board, so there are $(n+1)^2$ borders in total. We denote by b_L and b_R the borders corresponding to the left-hand and right-hand edges of the board respectively. Given two borders b_1 and b_2, we write $b_1 \leq b_2$ if and only if b_1 meets both the top and bottom of the board to the left of (or at the same point as) b_2, and write $b_1 < b_2$ if $b_1 \leq b_2$ and $b_1 \neq b_2$. Note that if $b_1 \leq b_2$ then b_1 lies entirely to the left of b_2 (the two borders may meet but never cross); this is a special property of $2 \times n$ boards and does not hold for edge-cuts in graphs corresponding to $k \times n$ boards for $k \geq 3$. Some examples of borders in a $2 \times n$ board are illustrated in Figure 6.1.

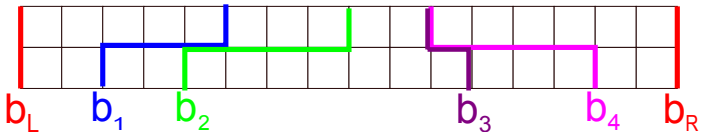


Figure 6.1: Examples of borders in a $2 \times n$ board, with $b_L < b_1 < b_2 < b_3 < b_4 < b_R$. Note that $B[b_1, b_3]$, $B[b_1, b_4]$, $B[b_2, b_3]$, $B[b_2, b_4]$ and $B[b_3, b_4]$ are sections, but $B[b_1, b_2]$ is not a section.

If G is the graph corresponding to the $2 \times n$ board B, we say that a vertex (or subgraph) is *incident* with a border b if the vertex (or some vertex in the subgraph) corresponds to a square on B whose edge forms part of b. If $b_1 < b_2$ are borders, we denote the subgraph induced by vertices lying between b_1 and b_2 by $B[b_1, b_2]$, and we say $B[b_1, b_2]$ is a *section* if it is connected.

Recall that, given any tree T, we denote by $\text{trunk}(T)$ the subtree obtained by deleting all leaves of T. Given any $x, y \in V(T)$ we set $P(T, x, y)$ to be the unique path from x to y in T.

6.1 Background and auxiliary results

Before describing our algorithm in the next section, we need a number of results which will be used to prove the algorithm's correctness. Recall the following theorem from Chapter 3, where $\mathcal{T}(G)$ denotes the set of all spanning trees of G.

Theorem (Theorem 3.1)**.** *Let G be a connected graph with colouring ω from colour-set C. Then, for any $d \in C$,*

$$m_G(G, \omega, d) = \min_{T \in \mathcal{T}(G)} m_T(T, \omega, d).$$

Recall also that, for any $d \in C$, we say that T is a *d-minimal* spanning tree for G with respect to ω if $m_T(T, \omega, d) = m_G(G, \omega, d)$. In the remainder of this section, we prove that in the special case in which G corresponds to a $2 \times n$ board, there is always a d-minimal spanning tree T such that trunk(T) is a path.

In doing so, we will also make use of Corollary 3.8, which shows that the number of moves required to flood a graph is bounded above by the sum of the numbers of moves required to flood connected subgraphs which cover the vertex-set.

Recall that a key step used to prove Theorem 3.1 was to prove Lemma 3.2, a special case of Corollary 3.8 where the underlying graph G is a tree and A and B are disjoint. In this section, we will need the following result, which is proved using an extension of part of the proof of Lemma 3.2 above.

Lemma 6.2. *Let T be a tree, with colouring ω from colour-set C, let A and B be disjoint subsets of $V(T)$ such that $V(T) = A \cup B$ and $T[A], T[B]$ are connected, and let x be the unique vertex of B with a neighbour in A. Suppose that*

- *the sequence S_A floods $T[A]$ with colour d_A,*
- *the sequence S_B floods $T[B]$ with colour d_B,*
- *at least one move of S_B changes the colour of x, and*

- *playing S_A in T changes the colour of x.*

Then

$$m_T(T, \omega, d_B) \leq |S_A| + |S_B|.$$

Proof. We proceed by induction on $|B|$. Note that we may assume without loss of generality that ω gives a proper colouring of B; otherwise we may contract monochromatic components. Suppose $|B| = 1$. Then S_A must change the colour of the only vertex in B (linking it to some $a \in A$), and so playing S_A in T makes the whole tree monochromatic with colour d_A. Thus $m_T(T, \omega, d_A) \leq |S_A|$, and

$$m_T(T, \omega, d_B) \leq m_T(T, \omega, d_A) + 1 \leq |S_A| + 1 \leq |S_A| + |S_B|,$$

as required, since by assumption $|S_B| \geq 1$.

Now suppose $|B| > 1$, so B is not monochromatic initially, and assume that the result holds for smaller B. Set ${S_B}'$ to be the initial segment of S_B, up to and including the move that first makes B monochromatic (in any colour d'), so any final moves that simply change the colour of B are omitted. We may, of course, have ${S_B}' = S_B$ (and so $d' = d_B$), if B is not monochromatic before the final move of S_B.

First suppose that S'_B does not change the colour of x (which is only possible in the case $S'_B \neq S_B$). Then playing S'_B in T to make B monochromatic cannot change the colour of any vertex in A, so if we play S'_B in T and then play S_A, this will still flood A with colour d_A. Moreover, as playing S'_B has not changed the colour of x, playing S_A will still change the colour of x, thus linking all of B to A and so flooding T with colour d_A. Hence, in this case, we have

$$m_T(T, \omega, d_A) \leq |S'_B| + |S_A|,$$

and so, as we must in this case have $|S'_B| < |S_B|$,

$$m_T(T, \omega, d_B) \leq 1 + m_T(T, \omega, d_A) \leq 1 + |S'_B| + |S_A| \leq |S_A| + |S_B|,$$

as required.

Suppose now that S'_B does change the colour of x. Before the final move of ${S_B}'$ there are $r \geq 2$ monochromatic components in B (all but one of which have colour d'), with vertex-sets $B_1, \ldots, B_r$. For $1 \leq i \leq r$, set S_i to be the subsequence of ${S_B}'$ consisting of moves played in B_i, and note that these subsequences partition ${S_B}'$. Observe also that playing S_i in $T[B_i]$ gives B_i colour d', so $m_{T[B_i]}(B_i, \omega, d') \leq |S_i|$.

Let B_1 be the unique component adjacent to A, and set $T_1 = T[A \cup B_1]$. Note that S_A floods $T_1[A]$ with colour d_A, and S_1 floods $T_1[B_1]$ with colour d'. Moreover, as playing S_A in T changes the colour of x, playing S_A in T_1 must also change the colour of x. Also, at least one move from S_B changes the colour of x, the unique vertex of B_1 with a neighbour in A, and this move must belong to S_1. Thus we can apply the inductive hypothesis to see that

$$m_{T_1}(T_1, \omega, d') \leq |S_A| + |S_1|.$$

Now suppose without loss of generality that B_2 is adjacent to B_1. We can then apply Corollary 3.8 to $T_2 = T[V(T_1) \cup B_2]$ to see that

$$m_{T_2}(T_2, \omega, d') \leq m_{T_1}(T_1, \omega, d') + m_{T[B_1]}(B_2, \omega, d') \leq |S_A| + |S_1| + |S_2|.$$

Continuing in this way, each time adding an adjacent component, we see that

$$m_T(T, \omega, d') \leq |S_A| + \sum_{i=1}^{r} |S_i| = |S_A| + |{S_B}'|.$$

Now, if $S_B{}' = S_B$, this immediately gives the desired result, as $d' = d_B$. Otherwise, note that $|S_B| \geq |S_B{}'| + 1$ and so

$$m_T(T, \omega, d_B) \leq m_T(T, \omega, d') + 1 \leq |S_A| + |S_B{}'| + 1 \leq |S_A| + |S_B|,$$

as required. □

In the next result, we exploit this lemma to give a strengthening of Corollary 3.8 under additional assumptions. This can be applied to show that, in certain situations, we may assume that *no* optimal sequence to flood a subtree can change the colour of any vertex outside the subtree, when played in a larger tree.

Proposition 6.3. *Let T be a tree, with colouring ω from colour-set C, and let X and Y be disjoint subtrees of T such that $T[V(X) \cup V(Y)]$ is connected, and such that*

- *there is a sequence S_X of k_1 moves that floods X with some colour $d' \in C$,*
- *there is a sequence S_Y of k_2 moves that floods Y with colour d, and that changes the colour of the unique neighbour of X in Y, and*
- *playing S_X in T changes the colour of at least one vertex in Y.*

Then, setting $T' = T \setminus (V(X) \cup V(Y))$, we have

$$m_T(T, \omega, d) \leq m_{T'}(T', \omega, d) + k_1 + k_2.$$

Proof. Let v be the unique neighbour of X in Y. Then S_X must change the colour of v, so we can apply Lemma 6.2 to see that

$$m_{T[V(X)\cup V(Y)]}(T[V(X) \cup V(Y)], \omega, d) \leq |S_X| + |S_Y| = k_1 + k_2.$$

Iterative application of corollary 3.8 then gives

$$\begin{aligned} m_T(T,\omega,d) &\leq m_{T[V(X)\cup V(Y)]}(T[V(X)\cup V(Y)],\omega,d) + m_{T'}(T',\omega,d) \\ &\leq m_{T'}(T',\omega,d) + k_1 + k_2, \end{aligned}$$

as required. □

Using the previous results, we are now ready to prove the key result of this section. Throughout this proof, we make use of the following observation, either explicitly or implicitly:

Observation. *Let S' be a sequence of moves played in a tree T', and suppose that T' is a subtree of a larger tree T. Then S' has the same effect on the vertices of T' whether it is played in T' or T.*

Lemma 6.4. *Let G with colouring ω (from colour-set C) be the graph corresponding to a $2 \times n$ flood-it board B, let H be a connected induced subgraph of G, and let u and w be vertices lying in the leftmost and rightmost columns of H respectively. Then, for any $d \in C$, there exists a d-minimal spanning tree T for H such that* $\text{trunk}(T) \subseteq P(T,u,w)$.

Proof. We proceed by induction on $m_H(H,\omega,d)$. Note that the result is trivially true if $m_H(H,\omega,d) = 0$ as the graph is initially monochromatic with colour d and so any spanning tree will do. Suppose then that $m_H(H,\omega,d) > 0$. Let S be an optimal sequence to flood H with colour d, and suppose that the last move of S is α.

If H is monochromatic in some colour $d' \in C$ before α is played, and so this final move just changes the colour of the whole graph to d, we see that $m_H(H,\omega,d') \leq m_H(H,\omega,d) - 1$. Thus we may apply the inductive hypothesis to obtain a d'-minimal spanning tree T for H such that $\text{trunk}(T) \subseteq P(T,u,w)$. But then

$$m_T(T,\omega,d) \leq 1 + m_T(T,\omega,d') = 1 + m_H(H,\omega,d') \leq m_H(H,\omega,d),$$

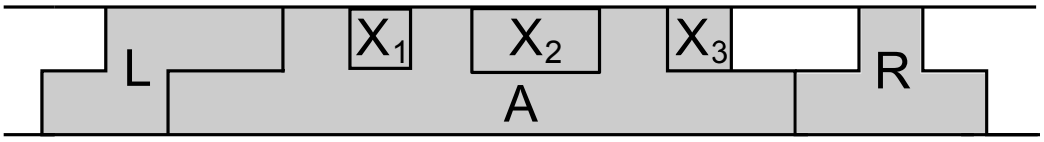


Figure 6.2: Monochromatic components of H before the final move is played.

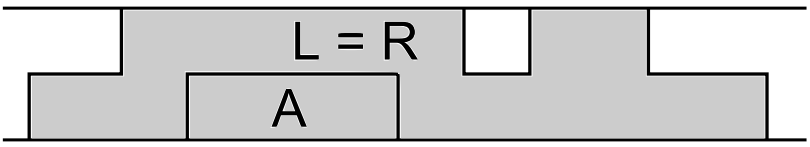


Figure 6.3: It is possible that $L = R$.

and so T is also a d-minimal spanning tree for H.

Thus we may assume that H is not monochromatic immediately before α is played. This means that α must change the colour of a monochromatic component A from some $d' \in C$ to d, where $H \setminus A$ is nonempty and has colour d before α is played. Since H is a connected induced subgraph of a $2 \times n$ board, $H \setminus A$ has at most one component L which contains vertices lying in columns to the left of all columns containing a vertex of A, and correspondingly at most one component R containing vertices lying in columns entirely to the right of A. There may additionally be some components $X_1, \ldots, X_r$ of $H \setminus A$ which contain only vertices which lie in the same column as some vertex of A. A possible structure for H is illustrated in Figure 6.2. We will exploit the structure of $H \setminus A$ to define a d-minimal spanning tree T for H whose non-leaf vertices lie on $P(T, u, w)$.

The remainder of the proof is structured as follows. We begin by defining a spanning tree T for H and identifying certain important substructures in T, and then go on to show that T is in fact d-minimal. To prove d-minimality, there are two cases, depending on whether $L \neq R$ (as in Figure 6.2) or $L = R$ (as in Figure 6.3); in the first of these cases, we will need to consider three subcases.

The Construction of T

Set v (respectively v') to be any vertex lying in the leftmost (respectively rightmost) column of A that has at least one neighbour in L (respectively R); if L (respectively R) is empty, we set $v = u$ (respectively $v' = w$). If $L \neq R$, and two vertices of L lie in the rightmost column of L, one of these must be adjacent to v, in which case we set this vertex to be u'; otherwise u' is defined to be the unique vertex of L that lies in the rightmost column. We define w' symmetrically in the case $L \neq R$, so that w' lies in the leftmost column of R, and if two vertices lie in this leftmost column of R then w' is adjacent to v'. If $L = R$ then we set $u' = w$ and $w' = u$.

Note that, since we can flood L with d by playing the sequence S but omitting the final move, we have $m_L(L, \omega, d) < m_H(H, \omega, d)$ and so, by the inductive hypothesis, there exists a d-minimal spanning tree T_L for L such that $\text{trunk}(T_L) \subseteq P(T_L, u, u')$. Similarly, there exists a d-minimal spanning tree T_R for R such that $\text{trunk}(T_R) \subseteq P(T_R, w', w)$, and also a d'-minimal spanning tree T_A for A such that $\text{trunk}(T_A) \subseteq P(T_A, v, v')$. Let S_A be an optimal sequence of moves to flood T_A with colour d', and S_L and S_R be optimal sequences to flood T_L and T_R respectively with colour d.

Observe that, as well as containing vertices that lie in columns to the left (respectively right) of A, L (respectively R) may additionally contain some vertices that lie in the same column as a vertex of A. We set T'_L to be the subtree of T_L induced only by those vertices in L that lie in the same column as or to the left of the leftmost vertex of A, and define T'_R symmetrically. Note that, even if $L = R$, we have $T'_L \cap T'_R = \emptyset$.

Now set T'_A to be the spanning tree for $H \setminus (T'_L \cup T'_R)$ obtained from T_A by adding an edge from every vertex z of this subgraph that does *not* lie in A to the vertex of A that lies in the same column as z (and observe that $\text{trunk}(T'_A) \subseteq P(T'_A, v, v')$). Assuming T_L and hence T'_L is nonempty, we define x to be the rightmost neighbour of v in T'_L; symmetrically, if $T_R \neq \emptyset$, we set y to be the leftmost vertex of T'_R that is adjacent to v' (note that if $T'_L = T_L$ we will have $x = u'$, and if $T'_R = T_R$ then $y = w'$).

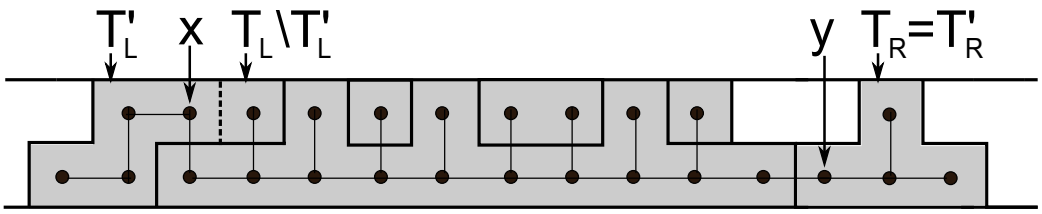


Figure 6.4: The spanning tree T.

We then obtain a spanning tree T for H by connecting T'_L, T'_A and T'_R with the edges xv and $v'y$. The construction of T is illustrated in Figure 6.4. It is clear from the construction that T is a spanning tree for H, and that $\text{trunk}(T) \subseteq P(T, u, w)$; we will argue that in fact T is a d-minimal spanning tree for H.

Case 1. $L \neq R$.

Having defined the spanning tree T for H, we now consider how to flood T with colour d in the case that $L \neq R$. First, observe that

$$\begin{aligned} |S| &\geq 1 + m_A(A, \omega, d') + m_L(L, \omega, d) + m_R(R, \omega, d) \\ &\qquad + \sum_{i=1}^{r} m_{X_i}(X_i, \omega, d) \\ &\geq 1 + |S_A| + |S_L| + |S_R| + |\bigcup_{i=1}^{r} \text{col}(X_i, \omega) \setminus \{d\}|. \end{aligned} \tag{6.1}$$

We now define S'_L (respectively S'_R) to be the subsequence of S_L (respectively S_R) consisting of moves that change the colour of at least one vertex in T'_L (respectively T'_R). Observe that, as $L \neq R$ and hence $S'_L \cap S'_R = \emptyset$, we may assume without loss of generality that all moves of S'_L (respectively S'_R) are in fact played in T'_L (respectively T'_R): any move of S'_L (respectively S'_R) that is not played here can be replaced with a move that is played in T'_L (respectively T'_R) and has exactly the same effect on T_L when played as part of S_L. Since T'_L (respectively T'_R) is a subtree of T_L (respectively T_R), any sequence of moves played in T'_L (respectively T'_R) will have the same effect on the vertices of T'_L (respectively T'_R) as when played in the larger tree T_L (respectively

T_R); thus, as S'_L (respectively S'_R) contains all moves of S_L (respectively S_R) that change the colour of vertices of T'_L (respectively T'_R), it must be that S'_L (respectively S'_R), played in T'_L (respectively T'_R) floods this tree with colour d. This immediately implies that $m_{T'_L}(T'_L, \omega, d) \leq |S'_L|$ and $m_{T'_R}(T'_R, \omega, d) \leq |S'_R|$.

Let C_L (respectively C_R) be the set of colours $\bar{d} \neq d$ such that a move of S_L (respectively S_R) is played in a monochromatic component of colour $\bar{d}$ that does not intersect T'_L (respectively T'_R). We also set $C_X = \bigcup_{i=1}^{r} \mathrm{col}(X_i, \omega) \setminus \{d\}$; we will call elements of $C_A = C_X \cup C_L \cup C_R$ *autonomous* colours. Note that, for each $z \in V(T_L) \setminus V(T'_L)$ that does not have colour d initially, at least one of the following must hold in order for z to be given colour d:

1. $\mathrm{col}(\{z\}, \omega) \in C_L$, or

2. either initially, or after some move of S_L, x has colour $\mathrm{col}(\{z\}, \omega)$.

Let W_L be the set of vertices $z \in V(T_L) \setminus V(T'_L)$ such that the first statement holds, so $z \in W_L$ if and only if $\mathrm{col}(\{z\}, \omega) \in C_L$. We then set U_L to be the set of vertices in $= V(T_L) \setminus V(T'_L)$ that do not belong to W_L and do not have colour d initially, and note that the second statement must hold for every $v \in U_L$. We can apply exactly the same reasoning to $V(T_R) \setminus V(T'_R)$ (replacing x with y), and define U_R and W_R analogously.

Observe that $|S_L| \geq |S'_L| + |C_L|$, and $|S_R| \geq |S'_R| + |C_R|$. Thus, by (6.1), we see that

$$|S| \geq 1 + |S_A| + |S'_L| + |S'_R| + |C_L| + |C_R| + |C_X| \tag{6.2}$$

$$\geq 1 + |S_A| + |S'_L| + |S'_R| + |C_A|. \tag{6.3}$$

We will consider three sub-cases, depending on whether none, one or both of U_L and U_R are nonempty. To summarise, the key properties of our construction in the case $L \neq R$ that we need when considering these subcases are as follows.

- The sequence S_A floods A with some colour.
- $V(T'_L) \setminus V(T_L)$ (respectively $V(T'_R) \setminus V(T_R)$) is covered by two sets U_L and W_L (respectively U_R and W_R).
- There is a sequence S'_L (respectively S'_R) that floods T'_L (respectively T'_R) with colour d and at some point gives x (respectively y) every colour in $\mathrm{col}(U_L, \omega)$ (respectively $\mathrm{col}(U_R, \omega)$).
- The sets $C_X = \bigcup_{i=1}^{r} \mathrm{col}(X_i, \omega) \setminus \{d\}$, $C_L = \mathrm{col}(W_L, \omega)$ and $C_R = \mathrm{col}(W_R, \omega)$ are such that

$$|S| \geq 1 + |S_A| + |S'_L| + |S'_R| + |C_L| + |C_R| + |C_X|.$$

These properties will enable us to prove in all three subcases that $m_T(T, \omega, d) \leq |S| = m_H(H, \omega, d)$, and hence that T is d-minimal.

Case 1.1. $U_L = U_R = \emptyset$.

In this case, we will first play S_A, flooding A, and then repeatedly change the colour of A to cycle through all colours in C_A. Note our first $|S_A| + |C_A|$ moves make T'_A monochromatic in some colour, so $m_{T'_A}(T'_A, \omega, d) \leq 1 + |S_A| + |C_A|$. Thus we can apply Corollary 3.8 to see that

$$\begin{aligned} m_T(T, \omega, d) &\leq m_{T'_L}(T'_L, \omega, d) + m_{T'_A}(T'_A, \omega, d) + m_{T'_R}(T'_R, \omega, d) \\ &\leq 1 + |S_A| + |S'_L| + |S'_R| + |C_A| \\ &\leq |S| && \text{by (6.3)} \\ &= m_H(H, \omega, d), \end{aligned}$$

as required.

Case 1.2. *Exactly one of U_L and U_R is nonempty.*

Without loss of generality suppose that $U_L \neq \emptyset$ (and hence that $U_R = \emptyset$). Once again, we begin by playing S_A and then changing the colour of A to cycle through all colours in C_A; these first $|S_A| + |C_A|$ moves create a monochromatic component A' containing $T'_A \setminus (U_L \cup U_R)$.

We now argue that we may assume that playing these moves in T does not change the colour of any vertex in T'_L. First, we claim that playing S_A in T cannot change the colour of any vertex in T'_L. Indeed, if this sequence does change the colour of a vertex in T'_L, it must change the colour of x, and this colour change will be due to moves in S_A changing the colour of v. Thus, if we played S_A in the tree T_1, obtained by connecting T_L, T_A and T_R with the edges xv and yv', v would go through the same sequence of colour changes and so the sequence would still change the colour of x (which is the unique vertex of T_L adjacent to T_A). However, as $U_L \neq \emptyset$, we know that playing S_L in T_L must change the colour of x (if there is a vertex in U_L that does not initially have the same colour as x, S_L must give x this colour; if every vertex in U_L starts off with the same colour as x then by definition this colour is not d, so again the colour of x must change as S_L is played), and so by Proposition 6.3 (setting $X = T_A$, $Y = T_L$, $S_X = S_A$ and $S_Y = S_L$) we would have $m_{T_1}(T_1, \omega, d) \leq |S_R| + |S_L| + |S_A| < |S|$, implying (by Theorem 3.1) that $m_H(H, \omega, d) \leq m_{T_1}(T_1, \omega, d) < |S| = m_H(H, \omega, d)$, a contradiction. Hence playing S_A in T will not change the colour of any vertex in T'_L.

We may further assume that then cycling A through all colours in C_A does not change the colour of any vertex in T'_L: if $\mathrm{col}(\{x\}, \omega) \in C_A$ we can choose this to be the last colour we play in A, and so our sequence will link A to x but will not change the colour of x (or therefore of any other vertex in T'_L). Thus we may indeed assume that the first $|S_A| + |C_A|$ moves do not change the colour of any vertex in T'_L.

Next, if playing S_A and cycling through the colours of C_A has not already linked A' to x, we play one further move to give A' the same colour as x. Since the sequence of moves we play up to this point does not change the colour of any vertex in T'_L,

we can now play the sequence S'_L to give every vertex in T'_L colour d. As x is in the same monochromatic component as A' this will also give all vertices in A' colour d. Moreover, playing this sequence will at some point give x, and hence A', every colour in $\mathrm{col}(U_L, \omega)$, and so will link every vertex in U_L to A' and ultimately give these vertices colour d. Thus, playing S_A, cycling through C_A, if necessary linking x to A', and then playing S'_L will flood all the vertices of $T \setminus T'_R$ with colour d (as $U_R = \emptyset$), so we see that

$$m_{T\setminus V(T'_R)}(T \setminus V(T'_R), \omega, d) \leq |S_A| + |C_A| + 1 + |S'_L|.$$

But then, once again, we can apply Corollary 3.8 to see that

$$\begin{aligned} m_T(T, \omega, d) &\leq m_{T\setminus V(T'_R)}(T \setminus V(T'_R), \omega, d) + m_{T'_R}(T'_R, \omega, d) \\ &\leq 1 + |S_A| + |C_A| + |S'_L| + |S'_R| \\ &\leq |S| && \text{by (6.3)} \\ &= m_H(H, \omega, d), \end{aligned}$$

as required.

Case 1.3. $U_L, U_R \neq \emptyset$.

In this final subcase, we begin by playing S_A, cycling A through all colours in $C_X \cup C_L$, and then (if required) playing an additional move to change the colour of the monochromatic component containing A to be the same as x; as before we may assume (choosing an appropriate order in which to cycle through the colours in $C_X \cup C_L$) that these initial moves do not change the colour of any vertex in T'_L.

Note that, as $U_L \neq \emptyset$, the colour of x must change at least once when we play S'_L in T'_L. Set β to be the last move in S'_L to change the colour of x, and note then that β must change the colour of some component Z, containing x, to d. Set $\bar{T_L} = T'_L \setminus V(Z)$,

and let S_Z be the subsequence of S'_L consisting of moves played in Z (so S_Z floods Z with colour d, and β is the final move of S_Z). As Z is monochromatic before β, playing $S_Z \setminus \beta$ in Z must flood this component with some colour $d_Z \in C$. Observe also that the sequence $S'_L \setminus S_Z$ must, when played in the forest $\bar{T}_L$, give every vertex of $\bar{T}_L$ colour d.

Suppose that, after playing S_A, cycling through $C_L \cup C_X$, and linking A to x, we then play $S_Z \setminus \beta$. This will ensure that x and hence A at some point receives every colour in $\mathrm{col}(U_L, \omega)$ (as $d \notin \mathrm{col}(U_L, \omega)$), so every vertex in U_L is linked to A. Note that we now have a monochromatic component B that contains A, Z and all vertices of $T'_A \setminus V(T_R)$ that do not initially have colour d.

We claim that the sequence of moves we play up to this point cannot change the colour of any vertex in T'_R. To prove the validity of this claim, set T_2 to be the spanning tree for H obtained by connecting T_R and $T \setminus V(T_R)$ with the edge yv'. It is clear that, if the sequence of moves we have played so far changes the colour of any vertex in T'_R when played in T, then playing the same sequence in T_2 would change the colour of $y \in T_R$ (v' will go through the same sequence of colour changes whether the moves are played in T or T_2, and vertices in T'_R will only change colour when the sequence is played in T if the colour changes of v' cause the colour of y to change). However, as $U_R \neq \emptyset$, we also know that S_R, played in T_R, changes the colour of y. Note also that all vertices of $T_2 \setminus (V(B) \cup V(T_R))$ that do not belong to $\bar{T}_L$ have colour d initially, so $m_{\bar{T}_L}(\bar{T}_L, \omega, d)$ moves suffice to flood $T_2 \setminus (V(B) \cup V(T_R))$ with colour d. We can now apply Proposition 6.3, setting S_X to be the sequence of moves we have

played up to this point, $X = B \setminus V(T_R)$, $Y = T_R$ and $S_Y = S_R$ to see that

$$\begin{aligned} m_{T_2}(T_2, \omega, d) &\leq m_{\bar{T}_L}(\bar{T}_L, \omega, d) + |S_A| + 1 + |S_Z| - 1 + |C_X| + |C_L| + |S_R| \\ &\leq |S'_L| - |S_Z| + |S_A| + |S_Z| + |C_X| + |C_L| + |S'_R| \\ &= |S'_L| + |C_L| + |S_R| + |S_A| + |C_X| \\ &< |S|. \end{aligned}$$

Theorem 3.1 would then imply that

$$m_H(H, \omega, d) \leq m_{T_2}(T_2, \omega, d) < |S| = m_H(H, \omega, d),$$

a contradiction.

Next we cycle the monochromatic component B through all colours in C_R (again, we may order these colours to ensure this does not change the colour of y); if this does not give B the same colour as y, we then play one further move to link this component to y. As we may therefore assume that all vertices in T'_R still have their initial colouring, if we now play S'_R, this will flood T'_R with colour d; as B and y lie in the same monochromatic component before these moves are played, this sequence will also give every vertex in B colour d. Moreover, linking B to y and playing S'_R will at some point give y, and hence A, every colour in $\mathrm{col}(U_R, \omega)$, and so all vertices in U_R will be linked to B and thus end up with colour d. So this sequence of moves gives every vertex in $T \setminus V(\bar{T}_L)$ colour d, and we have

$$\begin{aligned} m_{T \setminus V(\bar{T}_L)}(T \setminus V(\bar{T}_L), \omega, d) &\leq |S_A| + |C_X| + |C_L| + 1 + |S_Z| - 1 + |C_R| + 1 + |S'_R| \\ &\leq |S_A| + |C_A| + |S_Z| + |S'_R| + 1. \end{aligned}$$

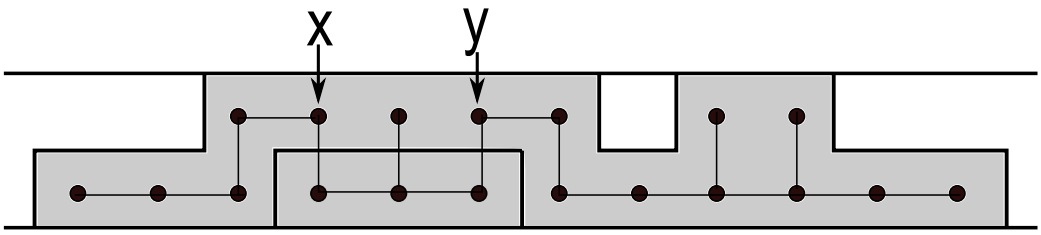


Figure 6.5: The construction of T in the case that $L = R$.

Finally, we apply Corollary 3.8 to give

$$\begin{aligned} m_T(T,\omega,d) &\leq m_{T\setminus V(\bar{T_L})}(T \setminus V(\bar{T_L}),\omega,d) + m_{\bar{T_L}}(\bar{T_L},\omega,d) \\ &\leq |S_A| + |C_A| + |S_Z| + |S'_R| + 1 + |S'_L| - |S_Z| \\ &= |S_A| + |C_A| + |S'_R| + |S'_L| + 1 \\ &\leq |S| \qquad \text{by (6.3)} \\ &= m_H(H,\omega,d), \end{aligned}$$

as required. This completes the proof in the final subcase for $L \neq R$.

Case 2. $L = R$.

For the case $L = R$, the structure of T is illustrated in Figure 6.5. Note that there cannot be any sets X_i in this case. The previous reasoning fails in the case $L = R$ because $S_L = S_R$ and so we may not be able to define disjoint subsequences S'_L and S'_R which flood T'_L and T'_R respectively. However, by considering more carefully the sequence of moves that floods $H \setminus V(A)$, we are able to deal with this problem.

If x and y belong to the same monochromatic component B of $T_L(= T_R)$ under the initial colouring ω (where this component has colour d_{xy}), then we can flood $\tilde{T} = T[V(B) \cup V(A)]$ by playing S_A and then changing the colour of A to d_{xy}: this implies that $m_{\tilde{T}}(\tilde{T},\omega,d_{xy}) \leq |S_A| + 1$. Let ω' be the colouring of T which agrees with ω on every vertex in T_L, and gives every vertex in A colour d_{xy}. Then T with colouring ω' is equivalent (when monochromatic components are contracted) to T_L

with colouring ω, implying that $m_T(T, \omega', d) = m_{T_L}(T_L, \omega, d) \leq |S_L|$. We can then apply Lemma 5.10 to give

$$\begin{aligned} m_T(T, \omega, d) &\leq m_T(T, \omega', d) + m_{T[V(T') \cup V(A)]}(T[V(T') \cup V(A)], \omega, d_{xy}) \\ &\leq |S_L| + |S_A| + 1 \\ &= |S| \\ &= m_H(H, \omega, d). \end{aligned}$$

So we may assume that x and y do not belong to the same monochromatic component initially. Let S' be the initial segment of S_L up to and including the move that first links x and y; let T' be the monochromatic component of $T \setminus V(A)$ that contains x and y at this point, suppose that T' has colour $\tilde{d}$ and that k moves of S' are played in T'.

We now claim that it suffices to prove that

$$m_{\tilde{T}}(\tilde{T}, \omega, \tilde{d}) \leq |S_A| + k + 1. \tag{6.4}$$

To see that this is indeed sufficient, set ω' to be the colouring of $V(T)$ that agrees with $S'(\omega, T_L)$ on T_L and gives all vertices of A colour $\tilde{d}$. Note that T with colouring ω' is equivalent (when monochromatic components are contracted) to T_L with colouring $S'(\omega, T_L)$, and so $m_T(T, \omega', d) = m_{T_L}(T_L, S'(\omega, T_L), d) \leq |S_L| - |S'|$. Let $\mathcal{A}$ be the set of monochromatic components of T with respect to ω', and suppose that each $A \in \mathcal{A}$ has colour c_A under this colouring. As k moves of S' are played in T', we can bound the number of moves required to give all the other monochromatic components with respect to ω' the colour they receive under ω':

$$\sum_{\substack{A \in \mathcal{A} \\ A \neq \tilde{T}}} m_A(A, \omega, c_A) \leq |S'| - k.$$

Thus

$$\sum_{A\in\mathcal{A}} m_A(A,\omega,c_A) \leq |S'| - k + m_{\tilde{T}}(\tilde{T},\omega,\tilde{d}),$$

and so, if (6.4) holds,

$$\sum_{A\in\mathcal{A}} m_A(A,\omega,c_A) \leq |S'| - k + |S_A| + k + 1 = |S'| + |S_A| + 1.$$

Lemma 5.10 then gives

$$\begin{aligned} m_T(T,\omega,d) &\leq m_T(T,\omega',d) + \sum_{A\in\mathcal{A}} m_A(A,\omega,c_A) \\ &\leq |S_L| - |S'| + |S'| + |S_A| + 1 \\ &= |S| \\ &= m_H(H,\omega,d), \end{aligned}$$

as required. Thus it is indeed sufficient to prove (6.4).

To prove the validity of (6.4), we will invoke the reasoning used in the case $L \neq R$; to do so we must show that appropriate versions of the key properties listed on page 100 hold in this case.

We begin with some definitions. Set $\tilde{T_L}$ to be the maximal monochromatic component of T' containing x immediately before the final move of S', and set $\tilde{T_R} = T' \setminus V(\tilde{T_L})$. Now let $\tilde{S_L}$ (respectively $\tilde{S_R}$) be the subsequence of S' consisting of moves played in $\tilde{T_L}$ (respectively $\tilde{T_R}$); note that $\tilde{S_L} \cap \tilde{S_R} = \emptyset$, $|\tilde{S_L}| + |\tilde{S_R}| = k$, and that playing $\tilde{S_L}$ (respectively $\tilde{S_R}$) in $\tilde{T_L}$ (respectively $\tilde{T_R}$) floods this tree with colour $\tilde{d}$. Further define $\tilde{T_L'}$ (respectively $\tilde{T_R'}$) to be the subtree of $\tilde{T_L}$ (respectively $\tilde{T_R}$) induced by vertices lying in the same column as or to the left (respectively right) of x (respectively y), and let $\tilde{S_L'}$ (respectively $\tilde{S_R'}$) be the subsequence of $\tilde{S_L}$ (respectively $\tilde{S_R}$) consisting of moves that change the colour of at least one vertex in $\tilde{T_L'}$ (respec-

tively $\tilde{T'_R}$). Note that, as $\tilde{S_L} \cap \tilde{S_R} = \emptyset$, we can in this case assume that every move of $\tilde{S'_L}$ (respectively $\tilde{S'_R}$) is in fact played in $\tilde{T'_L}$ (respectively $\tilde{T'_R}$), and hence that playing $\tilde{S'_L}$ in $\tilde{T'_L}$ (respectively $\tilde{S'_R}$ in $\tilde{T'_R}$) floods this subtree with colour $\tilde{d}$.

Now let $\tilde{C_L}$ (respectively $\tilde{C_R}$) be the set of colours $\bar{d} \neq \tilde{d}$ such that at least one move of $\tilde{S_L}$ (respectively $\tilde{S_R}$) is played in a monochromatic component of colour $\bar{d}$ that does not intersect $\tilde{T'_L}$ (respectively $\tilde{T'_R}$). Note that for every vertex $z \in V(\tilde{T_L}) \setminus V(\tilde{T'_L})$ that does not initially have colour $\tilde{d}$, at least one of the following must hold:

1. $\mathrm{col}(\{z\}, \omega) \in C_L$, or
2. either initially, or after some move of $\tilde{S_L}$, x has colour $\mathrm{col}(\{z\}, \omega)$.

Let $\tilde{W_L}$ be the set of vertices $z \in V(\tilde{T_L}) \setminus V(\tilde{T'_L})$ such that the first statement holds, and set $\tilde{U_L} = (V(\tilde{T_L}) \setminus V(\tilde{T'_L})) \setminus \tilde{W_L}$. Note that, for every $z \in \tilde{W_L}$, playing $\tilde{S'_L}$ in $\tilde{T'_L}$ must at some point give x colour $\mathrm{col}(\{z\}, \omega)$. We apply exactly the same reasoning to $V(\tilde{T_R}) \setminus V(\tilde{T'_R})$ (replacing x with y) and define $\tilde{U_R}$ and $\tilde{W_R}$ analogously. Observe that $|\tilde{S_L}| \geq |\tilde{S'_L}| + |\tilde{C_L}|$ and $|\tilde{S_R}| \geq |\tilde{S'_R}| + |\tilde{C_R}|$.

Thus our construction has the following properties.

- There is a sequence S_A which floods A with some colour.
- $V(\tilde{T_L}) \setminus V(\tilde{T'_L})$ (respectively $V(\tilde{T_R}) \setminus V(\tilde{T'_R})$) is partitioned into two sets, $\tilde{U_L}$ and $\tilde{W_L}$ (respectively $\tilde{U_R}$ and $\tilde{W_R}$).
- There is a sequence $\tilde{S'_L}$ (respectively $\tilde{S'_R}$) that floods $\tilde{T'_L}$ (respectively $\tilde{T'_R}$) with colour $\tilde{d}$ and at some point gives x (respectively y) every colour in $\mathrm{col}(U_L, \omega)$ (respectively $\mathrm{col}(U_R, \omega)$).
- The sets $\tilde{C_L} = \mathrm{col}(\tilde{U_L}, \omega)$ and $\tilde{C_R} = \mathrm{col}(\tilde{U_R}, \omega)$ are such that

$$k = |\tilde{S_L}| + |\tilde{S_R}| \geq |\tilde{S'_L}| + |\tilde{C_L}| + |\tilde{S'_R}| + |\tilde{C_R}|,$$

and hence

$$|S_A| + k + 1 \geq 1 + |S_A| + |\tilde{S'_L}| + |\tilde{S'_R}| + |\tilde{C_L}| + |\tilde{C_R}|.$$

As we do not have any set X_i in the case $L = R$, it follows that $\tilde{C_X} = \bigcup_{i=1}^{r} \mathrm{col}(X_i, \omega) \setminus \{d\} = \emptyset$, so we can equivalently write

$$|S_A| + k + 1 \geq 1 + |S_A| + |\tilde{S'_L}| + |\tilde{S'_R}| + |\tilde{C_L}| + |\tilde{C_R}| + |\tilde{C_X}|.$$

These properties correspond exactly to the list of properties on page 100 that are required for the proof in the case $L \neq R$; thus we can apply the same reasoning (with the three subcases) again to show that

$$m_{\tilde{T}}(\tilde{T}, \omega, \tilde{d}) \leq |S_A| + k + 1,$$

as required to demonstrate the validity of (6.4). This completes the final case of the proof. □

6.2 The algorithm

In this section we describe our algorithm to solve c-Free-Flood-It on $2 \times n$ boards, and use results from the previous section to prove its correctness.

We begin with some further definitions. For any section $B[b_1, b_2]$, we define $\mathcal{T}[b_1, b_2]$ to be the set of all spanning trees for $B[b_1, b_2]$. Given any $2 \times n$ Flood-It board B, corresponding to a graph G with colouring ω from colour-set C, we

define a set of vectors $Z(B)$, where

$$\begin{aligned} Z(B) = \{(b_1, b_2, r_1, r_2, d, I) : \\ & B[b_1, b_2] \text{ is a section,} \\ & r_1, r_2 \in V(B[b_1, b_2]), \\ & \exists T \in \mathcal{T}[b_1, b_2] \text{ such that trunk}(T) \subseteq P(T, r_1, r_2), \\ & r_1 \text{ incident with } b_1, r_2 \text{ incident with } b_2, \\ & d \in C, \\ & I \subseteq C\}. \end{aligned}$$

Note that there always exists a tree $T \in \mathcal{T}[b_1, b_2]$ such that trunk$(T) \subseteq P(T, r_1, r_2)$ unless one of the following holds:

1. there is more than one vertex of $B[b_1, b_2]$ lying strictly to the left of r_1 or strictly to the right of r_2, or

2. there is exactly one vertex of $B[b_1, b_2]$ lying strictly to the left of r_1 (respectively to the right of r_2), which is not adjacent to r_1 (respectively r_2) and whose neighbour in the same column as r_1 (respectively r_2) has no neighbour in $B[b_1, b_2]$ other than r_1 (respectively r_2).

Thus we can check whether this condition is satisfied in constant time.

We now introduce a function f which is closely related to the minimum number of moves required to flood a $2 \times n$ board. For any $\mathbf{z} = (b_1, b_2, r_1, r_2, d, I) \in Z(B)$ we define $f(\mathbf{z})$ to be the minimum, taken over all $T \in \mathcal{T}[b_1, b_2]$ such that trunk$(T) \subseteq P(T, r_1, r_2)$, of the number of moves that must be played in $P(T, r_1, r_2)$ to flood $P(T, r_1, r_2)$ with colour d, and link to $P(T, r_1, r_2)$ all leaves of T that do not have colours from I.

It follows immediately from Lemmas 6.4 and 5.9 that

$$m_G(G,\omega) = \min_{\substack{d\in C \\ r_1 \text{ incident with } b_L \\ r_2 \text{ incident with } b_R}} f(b_L, b_R, r_1, r_2, d, \emptyset).$$

Our algorithm in fact computes recursively a function f^*, with the same parameters as f. We will argue that, for every $\mathbf{z} \in Z(B)$, $f^*(\mathbf{z}) = f(\mathbf{z})$ and hence that it suffices to compute all values of f^* in order to calculate $m_G(G,\omega)$.

The first step of the algorithm is to initialise certain values of f^* to zero. We set $f^*(b_1, b_2, r_1, r_2, d, I) = 0$ if and only if, under the initial colouring, there exists a r_1-r_2 path of colour d in $B[b_1, b_2]$, and all vertices in $B[b_1, b_2]$ that do not lie on this path are adjacent to the path and have colours from $I \cup \{d\}$. All other values of $f^*(\mathbf{z})$ are initially set to infinity. Note that, under this definition, $f^*(\mathbf{z}) = 0 \iff f(\mathbf{z}) = 0$, and that for each $\mathbf{z} \in Z(B)$ we can easily determine in time $O(n)$ whether f^* should be initialised to zero or infinity.

In order to define further values of f^*, we introduce two more functions. First, for any $\mathbf{z} = (b_1, b_2, r_1, r_2, d, I) \in Z(B)$, we set

$$f_1(b_1, b_2, r_1, r_2, d, I) = 1 + \min_{d' \in C}\{f^*(b_1, b_2, r_1, r_2, d', I \cup \{d\})\}.$$

We also define, for any $\mathbf{z} \in Z(B)$,

$$f_2(b_1, b_2, r_1, r_2, d, I) = \min_{\substack{(b_1,b,r_1,x_1,d,I)\in Z(B) \\ (b,b_2,x_2,r_2,d,I)\in Z(B) \\ b_1<b<b_2 \\ x_1x_2\in E(G)}} \{f^*(b_1, b, r_1, x_1, d, I) + f^*(b, b_2, x_2, r_2, d, I)\}.$$

Finally, we set

$$f^*(\mathbf{z}) = \min\{f_1(\mathbf{z}), f_2(\mathbf{z})\}.$$

In Lemma 6.7 we will show how all values of f^* can be computed in the permitted time.

For the reasoning below, it will be useful to introduce another function θ, taking the same parameters as f and f^*. For any $\mathbf{z} = (b_1, b_2, r_1, r_2, d, I) \in Z(B)$, we define

$$\theta(\mathbf{z}) = f^*(\mathbf{z}) + |B[b_1, b_2]|.$$

In the following two lemmas, we show that $f^*(\mathbf{z}) = f(\mathbf{z})$ for all $\mathbf{z} \in Z(B)$, as claimed. We begin by demonstrating that $f^*(\mathbf{z})$ gives an upper bound for $f(\mathbf{z})$.

Lemma 6.5. *Let G with colouring ω (from colour-set C) be the coloured graph corresponding to a $2 \times n$ Flood-It board B. Then*

$$f(\mathbf{z}) \leq f^*(\mathbf{z})$$

for all $\mathbf{z} = (b_1, b_2, r_1, r_2, d, I) \in Z(B)$.

Proof. We proceed by induction on $\theta(\mathbf{z})$. Recall that we have equality between $f(\mathbf{z})$ and $f^*(\mathbf{z})$ whenever $f^*(\mathbf{z}) = 0$, so certainly the base case for $\theta(\mathbf{z}) = 0$ must hold. Assume therefore that $f^*(\mathbf{z}) > 0$, and that the result holds for all $\mathbf{z}'$ with $\theta(\mathbf{z}') < \theta(\mathbf{z})$.

Since $f^*(\mathbf{z}) > 0$, we must have $f^*(\mathbf{z}) \in \{f_1(\mathbf{z}), f_2(\mathbf{z})\}$. Suppose first that $f^*(\mathbf{z}) = f_1(\mathbf{z})$. Then, for some $d' \in C$,

$$\begin{aligned} f^*(b_1, b_2, r_1, r_2, d, I) &= 1 + f^*(b_1, b_2, r_1, r_2, d', I \cup \{d\}) \\ &\qquad \text{by definition of } f_1 \\ &\geq 1 + f(b_1, b_2, r_1, r_2, d', I \cup \{d\}) \\ &\qquad \text{by inductive hypothesis.} \end{aligned}$$

But then we know, by definition of f, that there exists $T \in \mathcal{T}[b_1, b_2]$ and a sequence

S of $f(b_1, b_2, r_1, r_2, d', I \cup \{d\})$ moves, all played in $P(T, r_1, r_2)$, which, when played in T, floods $P(T, r_1, r_2) \supseteq \text{trunk}(T)$ with colour d' and links all leaves to $P(T, r_1, r_2)$ except possibly those with colours from $I \cup \{d\}$. By appending one further move to S, which changes the colour of $P(T, r_1, r_2)$ to d, we obtain a sequence S' of length $f(b_1, b_2, r_1, r_2, d', I \cup \{d\}) + 1$ (with all moves played in $P(T, r_1, r_2)$) which, when played in T, floods $P(T, r_1, r_2)$ with colour d and is such that all leaves of T not linked to $P(T, r_1, r_2)$ by S have colours from I. Hence $f(b_1, b_2, r_1, r_2, d, I) \leq |S'| = 1 + f(b_1, b_2, r_1, r_2, d', I \cup \{d\})$, and so $f(b_1, b_2, r_1, r_2, d, I) \leq f^*(b_1, b_2, r_1, r_2, d, I)$, as required.

Now suppose that $f^*(\mathbf{z}) = f_2(\mathbf{z})$. Then, by definition of f_2, there must exists a border b with $b_1 < b < b_2$, and an edge $x_1x_2 \in E(G)$, such that (b_1, b, r_1, x_1, d, I), $(b, b_2, x_2, r_2, d, I) \in Z(B)$ and

$$f^*(b_1, b_2, r_1, r_2, d, I) = f^*(b_1, b, r_1, x_1, d, I) + f^*(b, b_2, x_2, r_2, d, I).$$

Note that $|B[b_1, b]|$ and $|B[b, b_2]|$ are both strictly smaller than $|B[b_1, b_2]|$, so by the inductive hypothesis we have

$$f^*(b_1, b_2, r_1, r_2, d, I) \geq f(b_1, b, r_1, x_1, d, I) + f(b, b_2, x_2, r_2, d, I).$$

By definition of f, there exist trees $T_1 \in \mathcal{T}[b_1, b]$ and $T_2 \in \mathcal{T}[b, b_2]$, and sequences S_1 and S_2 of length $f(b_1, b, r_1, x_1, d, I)$ and $f(b, b_2, x_2, r_2, d, I)$ respectively, such that (for $i \in \{1, 2\}$) all moves of S_i are played in $P(T_i, r_i, x_i)$ and S_i floods $P(T_i, r_i, x_i)$ with colour d, additionally linking all leaves of T_i to $P(T_i, r_i, x_i)$ except possibly those with colours from I. Now set $T = T_1 \cup T_2 \cup \{x_1x_2\}$. It is clear that $T \in \mathcal{T}[b_1, b_2]$, and moreover that $\text{trunk}(T) \subseteq P(T, r_1, r_2)$. Suppose T_1' and T_2' are the subtrees of T_1 and T_2 respectively that are given colour d by S_1 and S_2, and set $T' = T_1' \cup T_2' \cup \{x_1x_2\}$. Note that $P(T, r_1, r_2) \subseteq T'$ and that $\text{col}(T \setminus T', \omega) = \text{col}(T_1 \setminus T_1', \omega) \cup \text{col}(T_2 \setminus T_2', \omega) \subseteq I$,

so $f(b_1, b_2, r_1, r_2, d, I) \leq m_{T'}(T', \omega, d)$. We can then apply Corollary 3.8 to see that

$$\begin{aligned} f(b_1, b_2, r_1, r_2, d, I) &\leq m_{T'}(T', \omega, d) \\ &\leq m_{T'_1}(T'_1, \omega, d) + m_{T'_2}(T'_2, \omega, d) \\ &\leq |S_1| + |S_2| \\ &= f(b_1, b, r_1, x_1, d, I) + f(b, b_2, x_2, r_2, d, I) \\ &\leq f^*(b_1, b_2, r_1, r_2, d, I), \end{aligned}$$

completing the proof. □

Next we show that the reverse inequality also holds.

Lemma 6.6. *Let G with colouring ω (from colour-set C) be the coloured graph corresponding to a $2 \times n$ Flood-It board B. Then*

$$f(\mathbf{z}) \geq f^*(\mathbf{z})$$

for all $\mathbf{z} = (b_1, b_2, r_1, r_2, d, I) \in Z(B)$.

Proof. We proceed by induction on $f(\mathbf{z})$, noting again that we have equality in the base case for $f(\mathbf{z}) = 0$. Suppose that $f(\mathbf{z}) > 0$, and that the result holds for $\mathbf{z}'$ whenever $f(\mathbf{z}') < f(\mathbf{z})$. By definition, there exists a tree $T \in \mathcal{T}[b_1, b_2]$ and a sequence S of length $f(b_1, b_2, r_1, r_2, d, I)$ such that $\text{trunk}(T) \subseteq P(T, r_1, r_2)$, all moves of S are played in $P(T, r_1, r_2)$, and S floods $P(T, r_1, r_2)$ with colour d, leaving only leaves with colours from I not linked to $P(T, r_1, r_2)$. We proceed by case analysis on α, the final move of S.

Suppose first that $P(T, r_1, r_2)$ is already monochromatic before α, and that this final move just changes its colour to d from some $d' \in C$ (possibly flooding some additional leaves of colour d in the process). In this case it is clear that $f(b_1, b_2, r_1, r_2, d', I \cup$

$\{d\}) \leq |S| - 1$ and so we can apply the inductive hypothesis to see that

$$f^*(b_1, b_2, r_1, r_2, d', I \cup \{d\}) \leq f(b_1, b_2, r_1, r_2, d', I \cup \{d\}).$$

But then, by definition of f_1, we know that

$$\begin{aligned} f^*(b_1, b_2, r_1, r_2, d, I) &\leq 1 + f^*(b_1, b_2, r_1, r_2, d', I \cup \{d\}) \\ &\leq 1 + f(b_1, b_2, r_1, r_2, d', I \cup \{d\}) \\ &\leq 1 + |S| - 1 \\ &= f(b_1, b_2, r_1, r_2, d, I), \end{aligned}$$

as required.

So we may assume that $P(T, r_1, r_2)$ is not monochromatic before α: it may have either two or three monochromatic components. Suppose first that $P(T, r_1, r_2)$ has exactly three monochromatic components before α is played, A_1, A_2 and A_3; we may assume that A_1 and A_3 have colour d before α, and that this final move gives A_2 colour d to flood the entire path. For $i \in \{1, 2, 3\}$, set S_i to be the subsequence of $S \setminus \alpha$ consisting of moves played in A_i, and set $\bar{A}_i$ to be A_i together with all leaves of T that lie in the same column as a vertex of A_i or whose only neighbour on $P(T, r_1, r_2)$ is in A_i. Note that $\bar{A}_1$, $\bar{A}_2$ and $\bar{A}_3$ partition the vertex set of T, and that S_1, S_2 and S_3 partition $S \setminus \alpha$. We may assume without loss of generality that $r_1 \in A_1$ and $r_2 \in A_3$. Observe that there must exist borders b and b', with $b_1 < b < b' < b_2$, such that $\bar{A}_1 = B[b_1, b]$, $\bar{A}_2 = B[b, b']$ and $\bar{A}_3 = B[b', b_2]$. Set x_1x_2 to be the edge of T such that $x_1 \in A_1$, $x_2 \in A_2$ and y_1y_2 the edge of T such that $y_1 \in A_2$ and $y_2 \in A_3$.

Note that $T[A_1] \in \mathcal{T}[b_1, b]$, $T[A_2] \in \mathcal{T}[b, b']$, and $T[A_3] \in \mathcal{T}[b', b_2]$, and moreover that we have $\text{trunk}(T[A_1]) \subseteq P(T[A_1], r_1, x_1)$, $\text{trunk}(T[A_2]) \subseteq P(T[A_2], x_2, y_1)$ and $\text{trunk}(T[A_3]) \subseteq P(T[A_3], y_2, r_2)$. Observe also that S_1 is a sequence of moves

played in $P(T[A_1], r_1, x_1)$ that floods $P(T[A_1], r_1, x_1)$ with colour d and links all leaves, except possibly those with colours from I, to $P(T[A_1], r_1, x_1)$, so we must have $f(b_1, b, r_1, x_1, d, I) \leq |S_1|$. Similarly, we see that $f(b', b_2, y_2, r_2, d, I) \leq |S_3|$ and $f(b, b', x_2, y_1, d', I \cup \{d\}) \leq |S_2|$. Since $|S_1|, |S_2|, |S_3| < |S|$, we can apply the inductive hypothesis to see that

$$f^*(b_1, b, r_1, x_1, d, I) \leq f(b_1, b, r_1, x_1, d, I) \leq |S_1|$$

and

$$f^*(b', b_2, y_2, r_2, d, I) \leq f(b', b_2, y_2, r_2, d, I) \leq |S_3|.$$

The inductive hypothesis also gives

$$f^*(b, b', x_2, y_1, d', I \cup \{d\}) \leq f(b, b', x_2, y_1, d', I \cup \{d\}) \leq |S_2|,$$

and we can then apply the definition of f_1 to see that

$$f^*(b, b', x_2, y_1, d, I) \leq 1 + f^*(b, b', x_2, y_1, d', I \cup \{d\}) \leq 1 + |S_2|.$$

Now we can apply the definition of f^* to see that

$$\begin{aligned} f^*(b_1, b_2, r_1, r_2, d, I) &\leq f_2(b_1, b_2, r_1, r_2, d, I) \\ &\leq f^*(b_1, b, r_1, x_1, d, I) + f^*(b, b_2, x_2, r_2, d, I) \\ &\leq f^*(b_1, b, r_1, x_1, d, I) + f_2(b, b_2, x_2, r_2, d, I) \\ &\leq f^*(b_1, b, r_1, x_1, d, I) + f^*(b, b', x_2, y_1, d, I) \\ &\qquad + f^*(b', b_2, y_2, r_2, d, I) \\ &\leq 1 + |S_1| + |S_2| + |S_3| \\ &= |S| \\ &= f(b_1, b_2, r_1, r_2, d, I), \end{aligned}$$

as required.

For the remaining case, in which $P(T, r_1, r_2)$ has exactly two monochromatic components before α, we can use the same reasoning as in the previous case for three components to show that we must once again have $f^*(b_1, b_2, r_1, r_2, d, I) \leq f(b_1, b_2, r_1, r_2, d, I)$, completing the proof. □

The final step to is to show that all values of f^* can be computed in time $O(n^{10}2^c)$.

Proposition 6.7. *For any $2\times n$ Flood-It board B, the function $f^*(\mathbf{z})$ can be computed, for all $\mathbf{z} \in Z(B)$, in time $O(n^{10}2^c)$.*

Proof. We compute values of f^* recursively using a dynamic programming technique. Our table has one entry for each pair of borders, for each possible vertex incident with each of the borders, for each colour in the colour-set and for each possible subset of colours, so the total number of entries is at most

$$O(n^2 \cdot n^2 \cdot n \cdot n \cdot c \cdot 2^c) = O(n^6 c 2^c).$$

The table is initialised by setting all values to either zero or infinity, and for each entry we can determine which of these values it should take in time at most $O(n)$, so we can initialise the entire table in time $O(n^7 c 2^c)$.

The next step is to apply the recursive definition of f^* repeatedly to all entries in the table that are not already set to zero. Each time we apply this definition to a single entry, we take the minimum of at most $O(c + n^3)$ values (one for each choice of colour, plus one for each combination of a border and a pair of adjacent vertices on either side), each a combination of at most two other entries in the table, so each entry can be calculated in time $O(c + n^3)$. We can therefore perform one iteration in which we apply the definition to each non-zero entry in the table in time $O(n^9 2^c)$.

Note that once we have initialised the table, we have the correct value of $f^*(\mathbf{z})$ for any $\mathbf{z}$ such that $\theta(\mathbf{z}) = 0$. Moreover, the value of $f^*(\mathbf{z})$ depends only on values of $f^*(\mathbf{z}')$ where $\theta(\mathbf{z}') < \theta(\mathbf{z})$, so after k iterations we will have correctly computed the value of $f^*(\mathbf{z})$ for all $\mathbf{z}$ with $\theta(\mathbf{z}) \leq k$. Note that for every $\mathbf{z} \in Z(B)$, $\theta(\mathbf{z}) \leq 4n$, as there are at most $2n$ vertices lying between any pair of borders, and no more than $2n$ moves can be required to flood a graph with at most this many vertices, so $4n$ iterations are sufficient to guarantee we have computed all values of f^* correctly.

Thus, we can compute all values of $f^*(\mathbf{z})$ for $\mathbf{z} \in Z(B)$ in time $O(n^{10} 2^c)$, as required. □

We now combine the previous three results to give the proof of this chapter's main theorem.

Proof of Theorem 6.1. Recall that, from the definition of f and Lemmas 6.4 and 5.9,

$$m_G(G, \omega) = \min_{\substack{d \in C \\ r_1 \text{ incident with } b_L \\ r_2 \text{ incident with } b_R}} f(b_L, b_R, r_1, r_2, d, \emptyset).$$

Thus, in order to compute $m_G(G, \omega)$ in time $O(n^{10} 2^c)$, it suffices to compute all

relevant values of f in time $O(n^{10}2^c)$.

However, we know from Lemmas 6.5 and 6.6 that $f(\mathbf{z}) = f^*(\mathbf{z})$ for all $\mathbf{z} \in Z(B)$, and from Proposition 6.7 we know that we can compute $f^*(\mathbf{z})$ for all $\mathbf{z} \in Z(B)$ in time $O(n^{10}2^c)$. This completes the proof of the theorem. □

Chapter 7

The Parameterised Complexity of List Problems on Graphs of Bounded Treewidth

In the previous chapter we considered the parameterised complexity of c-Free-Flood-It (with parameter c), a problem that we had shown to be NP-hard in general. In this chapter we again consider the parameterised complexity of NP-hard problems, but move the focus towards some more classical graph colouring problems, and take the treewidth of the input graph as the parameter. We will make use of the definitions of treewidth and of various types of graph colouring introduced in Sections 1.3.1 and 1.2 respectively.

Many graph problems that are known to be NP-hard in general are fixed parameter tractable when parameterised by the treewidth k of the graph, that is they can be solved in time $f(k) \cdot n^{O(1)}$ for some function f. Often there even exist linear-time algorithms to solve such problems on graphs of fixed treewidth [6, 14, 49]. This is the case for a number of graph colouring problems. Although it is NP-hard to determine whether an arbitrary graph is 3-colourable, the chromatic number of a graph

of treewidth at most k can be computed in linear time (Arnborg and Proskurowski [6]). Similarly, while it is NP-hard to determine the edge-chromatic number of cubic graphs (Holyer [32]) and more generally d-regular graphs for any $d \geq 3$ (Leven and Galil [41]), this problem can again be solved in linear time on graphs of bounded treewidth (Zhou, Nakano and Nishizeki [53]). The pattern is the same for the total colouring problem: this is NP-hard even for regular bipartite graphs (McDiarmid and Sánchez-Arroyo [43]) but there exists a linear-time algorithm to solve the problem on graphs of bounded treewidth (Isobe, Zhou and Nishizeki [33]).

However, list versions of such problems cannot always be solved so efficiently on graphs of bounded treewidth. The well-known list variant of vertex colouring is clearly NP-hard in general (as it generalises the standard colouring problem), but it can also be solved in time $O(|V|^{k+2})$ on graphs of fixed treewidth k, and even in linear time on such graphs if the number of colours is bounded (Jansen and Scheffler [34]). However, in contrast with the linearity result for Chromatic Number on graphs of fixed treewidth, it has been shown that, when the number of colours is unbounded, List Colouring is W[1]-hard and so not (up to certain complexity theoretic assumptions) fixed parameter tractable (Fellows, Fomin, Lokshtanov, Rosamond, Saurabh, Szeider and Thomassen [23]). The list variants of edge-colouring and total colouring are harder still on graphs of bounded treewidth: both problems are NP-hard on series-parallel graphs (Zhou, Matsuo and Nishizeki [52]), which have treewidth at most two, and list edge-colouring is also NP-hard on outerplanar graphs (Marx [42]), another class of graphs with treewidth at most two (in both cases the reductions require the colour-lists to be drawn from a set of unbounded size).

Perhaps surprisingly, however, it can be easier to determine the minimum length of lists required to guarantee the existence of a proper list colouring for a graph G than to determine whether G admits a proper colouring with a particular set of lists. Alongside the negative result about the complexity of List Colouring, in [23] the

authors also use Courcelle's theorem [18] to prove the following result.

Theorem 7.1 ([23])**.** *The* List Chromatic Number *problem, parameterised by the treewidth bound t, is fixed-parameter tractable, and solvable in linear time for any fixed t.*

In Section 7.1, we prove two further surprising results of this nature, showing that the same pattern extends to the list chromatic number problems for edge-colouring and total colouring. We consider the following problems.

List Edge Chromatic Number
Input: A graph $G = (V, E)$.
Question: What is $\mathrm{ch}'(G)$?

List Total Chromatic Number
Input: A graph $G = (V, E)$.
Question: What is $\mathrm{ch}_T(G)$?

Determining whether there exists a proper list edge or list total colouring with a given set of lists is known to be NP-hard even on graphs of treewidth two, and yet we demonstrate that List Edge Chromatic Number and List Total Chromatic Number can be solved in linear time on graphs of bounded treewidth. This answers an open question of Fellows et. al. [23]. The structural theorems we use to prove these results are in fact special cases of the famous List (Edge) Colouring Conjecture and Total Colouring Conjecture respectively (discussed on pages 8 and 9).

In Section 7.2, we go on to consider a list version of the Hamilton Path problem, giving another example of an NP-hard problem that is solvable in linear time on graphs of bounded treewidth but has a W[1]-hard list version.

7.1 List Colouring Problems

The main result of this section is the following theorem.

Theorem 7.2. List Edge Chromatic Number *and* List Total Chromatic Number *are fixed parameter tractable, parameterised by the treewidth bound k, and are solvable in linear time for any fixed k.*

The key technical tools we use to prove this result are Theorems 7.7 and 7.8, which determine the list edge chromatic number and list total chromatic number respectively for graphs with fixed treewidth and large maximum degree: under suitable conditions on the maximum degree, the list edge chromatic number is equal to the maximum degree, and the list total chromatic number is equal to the maximum degree plus one.

Using Theorems 7.7 and 7.8, we see that it suffices to solve List Edge Chromatic Number and List Total Chromatic Number in the case that the maximum degree is bounded. Recall that solving these problems is equivalent to solving the vertex-colouring version, List Chromatic Number, on the line or total graph respectively, and that we know from Theorem 7.1 that List Chromatic Number can be solved in linear time on graphs of bounded treewidth. However, in general the treewidth of $L(G)$ or $T(G)$ can be arbitrarily large, even when the treewidth of G is small, so results about the parameterised complexity of vertex colouring problems do not immediately transfer to the edge and total colouring cases.

In the case that the maximum degree of G, as well as the treewidth of G, is bounded, however, the following result (proved in [19]) tells us that the treewidth of $L(G)$ is bounded by a constant multiple of that of G.

Lemma 7.3. *Let G be a graph of treewidth at most k, and maximum degree at most Δ. Then $L(G)$ has treewidth at most $(k+1)\Delta$.*

A similar result holds for the treewidth of $T(G)$.

Lemma 7.4. *Let G be a graph of treewidth at most k, and maximum degree at most Δ. Then $T(G)$ has treewidth at most $(k+1)(\Delta+1)-1$.*

Proof. If $(T, \mathcal{D})$ is a width k tree decomposition for G, it is easy to verify that $(T, \mathcal{D}')$, where $\mathcal{D}'(t) = \mathcal{D}(t) \cup \{uv \in E : \{u, v\} \cap \mathcal{D}(t) \neq \emptyset\}$, is a tree decomposition of $T(G)$ of width at most $(k+1)(\Delta+1)-1$. □

Thus, when the maximum degree of G is bounded, we *can* apply Theorem 7.1 to $L(G)$ and $T(G)$ to show that List Edge Chromatic Number and List Total Chromatic Number can be solved in linear time on graphs of bounded treewidth.

We will need two further results for our proofs. First, we use a theorem of Galvin concerning the list edge chromatic number of bipartite graphs:

Theorem 7.5 ([28])**.** *Let G be a bipartite graph. Then*

$$\mathrm{ch}'(G) = \chi'(G) = \Delta(G).$$

We also make use of an algorithm of Bodlaender:

Theorem 7.6 ([11])**.** *For all $k \in \mathbb{N}$, there exists a linear-time algorithm that tests whether a given graph $G = (V, E)$ has treewidth at most k and, if so, outputs a tree-decomposition of G with treewidth at most k.*

We now prove our results about the list edge chromatic number and list total chromatic number of graphs with bounded treewidth and large maximum degree. We begin with the list edge chromatic number case, proving a special case of the List (Edge) Colouring Conjecture.

Theorem 7.7. *Let G be a graph with treewidth at most k and $\Delta(G) \geq (k+1)2^{k+2}$. Then $\mathrm{ch}'(G) = \Delta(G)$.*

Proof. As $\Delta \leq \mathrm{ch}'(G)$, it is sufficient to prove that, if G is a graph of treewidth at most k, we have $\mathrm{ch}'(G) \leq \max\{\Delta(G), (k+1)2^{k+2}\}$. Suppose this does not hold, and let G be a counterexample with as few edges as possible, so there exists a set $\mathcal{L} = (L_e)_{e \in E(G)}$ of colour-lists, all of length $\Delta_0 = \max\{\Delta(G), (k+1)2^{k+2}\}$, such that there is no proper list edge-colouring of $(G, \mathcal{L})$. We may assume without loss of generality that G contains no isolated vertices and so, by edge-minimality of G, we must have $\mathrm{ch}'(G') \leq \max\{\Delta(G'), (k+1)2^{k+2}\} \leq \Delta_0$ for any proper subgraph G' of G.

We begin by observing that every edge $e \in E(G)$ must be incident with at least Δ_0 others: by edge-minimality of G, there exists a proper list edge-colouring of $(G - e, \mathcal{L} \setminus \{L_e\})$, and if e is incident with fewer than Δ_0 other edges then we can certainly extend this to a proper list edge-colouring of $(G, \mathcal{L})$. Using this observation, we will show that there must exist a nonempty set of vertices $U \subsetneq V$ such that any proper list edge-colouring of $(G \setminus U, (L_e)_{e \in E(G \setminus U)})$ can be extended to a proper list edge-colouring of $(G, \mathcal{L})$, contradicting the choice of G as an edge-minimal counterexample (as the fact there are no isolated vertices in G means $|E(G \setminus U)| < |E(G)|$).

Let us define $L \subset V(G)$ to be the set of vertices of degree at least $\Delta_0/2$, and note that every edge is incident with at least one vertex from L (as otherwise it can be incident with only $\Delta_0 - 1$ other edges), so $S = V(G) \setminus L$ is an independent set. Fix a width k tree decomposition $(T, \mathcal{D})$ of G, and choose $v \in L$ such that $\mathrm{depth}(t_v) = \max_{x \in L} \mathrm{depth}(t_x)$. We then set T' to be the subtree of T rooted at t_v, that is the subgraph of T induced by nodes u such that the path from u to the root contains t_v.

Set $X \subseteq V(G)$ to be $\bigcup_{t \in T'} \mathcal{D}(t)$, and $X' = X \setminus \mathcal{D}(t_v)$. We then make the following observations.

1. $L \cap X \subseteq \mathcal{D}(t_v)$: if any vertex $z \in L$ appears in a bag indexed by T' but does not appear in $\mathcal{D}(t_v)$, we must have $\mathrm{depth}(t_z) > \mathrm{depth}(t_v)$, contradicting the choice

of v. This implies immediately that $X' \subseteq S$.

2. $\Gamma(X') \subseteq \mathcal{D}(t_v)$: no vertex from X' can appear in a bag of the decomposition not indexed by T', so clearly we have $\Gamma(X') \subseteq X$; but also, as $X' \subset S$, we have $\Gamma(X') \subseteq L$ and so we see $\Gamma(X') \subseteq L \cap X \subseteq \mathcal{D}(t_v)$.

3. $|X'| \geq \Delta_0/2 - k$: since v appears only in bags indexed by T', we have $\Gamma(v) \subseteq X \setminus \{v\}$, implying $|X| - 1 \geq d(v) \geq \Delta_0/2$, and we know $|X| - |X'| = |\mathcal{D}(t_v)| \leq k+1$, so we have $|X'| \geq |X| - k - 1 \geq \Delta_0/2 - k$.

As the neighbourhood of any vertex $x \in X'$ is contained in $\mathcal{D}(t_v)$, there are at most 2^{k+1} possibilities for the neighbourhood of such a vertex. Therefore there must exist some subset $U \subseteq X' \subseteq X$ such that every vertex in U has the same neighbourhood, and

$$\begin{aligned} |U| &\geq \frac{|X'|}{2^{k+1}} \\ &\geq \frac{\Delta_0/2 - k}{2^{k+1}} \\ &\geq \frac{(k+1)2^{k+1} - k}{2^{k+1}} \\ &\geq k+1 && \text{as } |U| \in \mathbb{N} \\ &\geq |\Gamma(U)|. \end{aligned}$$

Now let ϕ be a proper edge-colouring of $(G \setminus U, (L_e)_{e \in E(G \setminus U)})$. If we can extend ϕ to a proper edge-colouring of G in which every edge incident with U also receives a colour from its list, then we have a proper list edge-colouring of $(G, \mathcal{L})$, giving the required contradiction.

Set $W = \Gamma(U)$, say $W = \{w_1, \ldots, w_r\}$ (where $r \leq |U|$), and let H be the complete bipartite subgraph of G induced by $U \cup W$. Suppose, for $1 \leq i \leq r$, that F_i is the set of colours already used by ϕ on edges incident with w_i, and for each $u \in U$ define the list L'_{uw_i} to be $L_{uw_i} \setminus F_i$. If we can properly colour the edges of H in such a way that

each edge $e \in E(H)$ is given a colour from L'_e, then we can extend ϕ as required.

Observe that, for each i, $|F_i| \leq \Delta(G) - |U|$, and so we have $|L'_{uw_i}| \geq \Delta_0 - \Delta(G) + |U| \geq |U|$. But as H is bipartite, with maximum degree $|U|$, we have (by Theorem 7.5)

$$\mathrm{ch}'(H) = \Delta(H) = |U| \ .$$

Therefore, as each list L'_e contains at least $|U|$ colours, there exists a proper edge colouring of H in which every edge e receives a colour from its list L'_e, completing the proof. □

We use a very similar argument to prove the following result about the list total chromatic number, which is a special case of the Total Colouring Conjecture.

Theorem 7.8. *Let G be a graph with treewidth at most k and $\Delta(G) \geq (k+1)2^{k+2}$. Then $\mathrm{ch}_T(G) = \Delta(G) + 1$.*

Proof. As $\mathrm{ch}_T(G) \geq \Delta(G)+1$, it suffices to prove that, if G is a graph of treewidth at most k, we have $\mathrm{ch}_T(G) \leq \Delta_0+1 = \max\{\Delta(G), (k+1)2^{k+2}\}+1$. Again, we prove the result by contradiction, so suppose this does not hold, and let G be a counterexample with as few edges as possible, so there exists a set $\mathcal{L} = (L_x)_{x \in V(G) \cup E(G)}$ of colour-lists, all of length $\Delta_0 + 1$, such that there is no proper list total colouring of $(G, \mathcal{L})$. We may assume without loss of generality that G contains no isolated vertices and so, by edge-minimality of G, we must have $\mathrm{ch}_T(G') \leq \max\{\Delta(G'), (k+1)2^{k+2}\}+1 \leq \Delta_0+1$ for any proper subgraph G' of G.

We may assume that every edge $e \in E(G)$ is incident with at least $\Delta_0 - 1$ others: if e is incident with fewer than $\Delta_0 - 1$ other edges then at most Δ_0 colours can appear on vertices or edges incident with e, and so any proper list total colouring of $(G - e, \mathcal{L} \setminus \{L_e\})$ can be extended to a proper list total colouring of $(G, \mathcal{L})$. Thus, if we define $L' \subset V(G)$ to be the set of vertices of degree at least $(\Delta_0 - 1)/2$, every edge

is incident with at least one vertex from L', and $S' = V(G) \setminus L'$ is an independent set.

Exactly as in the proof of Theorem 7.7, we can find a subset $U \subset S'$ such that all vertices of U have the same neighbourhood of size at most $k+1$, and

$$\begin{aligned} |U| &\geq \frac{(\Delta_0 - 1)/2 - k}{2^{k+1}} \\ &\geq \frac{((k+1)2^{k+2} - 1)/2 - k}{2^{k+1}} \\ &> k+1 && \text{as } |U| \in \mathbb{N} \\ &\geq |\Gamma(U)|. \end{aligned}$$

As $G \setminus U$ is a proper subgraph of G, there exists a proper list total colouring ϕ of $(G \setminus U, \mathcal{L}')$ where $\mathcal{L}' = (L_x)_{x \in E(G\setminus U) \cup V(G \setminus U)}$. If we can extend ϕ to a proper total colouring of G in which every vertex from U and every edge incident with U also receives a colour from its list, then we have a proper list total colouring of $(G, \mathcal{L})$, giving the desired contradiction.

As in the proof of Theorem 7.7, set $W = \Gamma(U)$, say $W = \{w_1, \ldots, w_r\}$ (where $r \leq |U|$), and let H be the complete bipartite subgraph of G induced by $U \cup W$. Suppose, for $1 \leq i \leq r$, that F_i is the set of colours already used by ϕ on w_i and the edges incident with w_i, and for each $u \in U$ define L'_{uw_i} to be $L_{uw_i} \setminus F_i$.

Observe that, for each i, $|F_i| \leq 1 + \Delta(G) - |U|$, and so we have

$$|L_{uw_i}| \geq \Delta_0 + 1 - 1 - \Delta(G) + |U| \geq |U|.$$

As H is bipartite, with maximum degree $|U|$, we have (by Theorem 7.5) $\mathrm{ch}'(H) = \Delta(H) = |U|$. Therefore, as each list L'_e contains at least $|U|$ colours that are not already used by ϕ on edges or vertices incident with e, we can extend ϕ to a proper list colouring ϕ' including the edges incident with U. If we can then colour the vertices

of U in such a way that no $u \in U$ receives a colour used by ϕ' on an adjacent vertex or incident edge, we can indeed extend ϕ to a proper list total colouring of G.

Note that every $u \in U$ has degree at most $k+1$, and so is adjacent to or incident with at most $2(k+1)$ vertices and edges of G. Thus, as $|L_u| \geq \Delta_0 + 1 > 2(k+1)$, there is at least one colour in L_u that is not used by ϕ' on a vertex adjacent to u or on an edge incident with u, and we can extend ϕ' to the vertices of U greedily. This gives a proper list total colouring of $(G, \mathcal{L})$, completing the proof. □

Using the two previous theorems, we are now ready to prove Theorem 7.2, the surprising complexity result that states that List Edge Chromatic Number and List Total Chromatic Number are fixed parameter tractable when parameterised by treewidth, and are solvable in linear time on graphs of any fixed treewidth.

Proof of Theorem 7.2. Let G be a graph on n vertices, with treewidth at most k, and set $f(k) = (k+1)2^{k+2}$. We can check in time $O(f(k)n)$ whether $\Delta(G) \geq f(k)$, and if this is the case then, by Theorem 7.7 or Theorem 7.8, we know the exact value of $\mathrm{ch}'(G)$ or $\mathrm{ch}_T(G)$ respectively. Thus it suffices to solve both problems in the case that $\Delta(G) < f(k)$.

This is exactly the same as solving List Chromatic Number on $L(G)$ or $T(G)$, when $\Delta(G) < f(k)$. But in this case, by Lemmas 7.3 and 7.4, $L(G)$ and $T(G)$ have bounded treewidth. Note that both graphs can be computed from G in time

$$\begin{aligned} O(|E(L(G))| + |E(T(G))|) &= O((k+1)(f(k)+1)(|T(G)| + |L(G)|) \\ &= O(k^2 f(k)|G|), \end{aligned}$$

and so we can then use Bodlaender's algorithm (Theorem 7.6) to find a tree decomposition of $L(G)$ or $T(G)$, of width at most $(k+1)(f(k)+1)$, in time $O(|G|)$ for any fixed k. By Theorem 7.1, given this decomposition, we can compute the list chromatic number of $L(G)$ or $T(G)$, and hence the list edge chromatic number or list

total chromatic number of G, in linear time. □

Our proof of Theorem 7.7 also gives rise to a polynomial-time algorithm to compute a proper list edge-colouring of $(G, \mathcal{L})$ for any graph G of fixed treewidth and large maximum degree, provided every $L \in \mathcal{L}$ has length at least $\Delta(G)$. Correspondingly, we can exploit the proof of Theorem 7.8 to compute a list total colouring of such a graph, provided every list has length at least $\Delta(G) + 1$. The idea is identical for both algorithms, so we only describe the algorithm to compute a proper list edge colouring.

Theorem 7.9. *Let G be a graph of order n and treewidth k, with maximum degree at least $(k+1)2^{k+3}$, and let $\mathcal{L} = (L_e)_{e \in E(G)}$ be a set of colour-lists such that $|L_e| \geq \Delta(G)$ for all $e \in E(G)$. Then, for fixed k, we can compute a proper list edge-colouring of $(G, \mathcal{L})$ in time $O(n^2)$.*

Proof. The idea of the algorithm is to delete repeatedly edges or sets of vertices, as in the proof of Theorem 7.7, until the degree of the remaining graph is less than $(k+1)2^{k+2} < \Delta(G)/2$, so the remaining edges can be list-coloured greedily. Edges and vertices are then reinserted and coloured to extend this colouring to the edges of G. Unlike in the proof of Theorem 7.7, where we identified a set of *at least* $k+1$ vertices to delete, in this algorithm we always delete a set of *exactly* $k+1$ vertices.

We begin with some preprocessing. For each vertex, we construct a list of its neighbours, and we also construct an n-element array storing the degree of each vertex. For vertices of degree at most $k+1$, the list of neighbours is sorted into order, and the list of vertices of degree at most $k+1$ is also sorted, in order of neighbourhoods (so that vertices with the same neighbourhood occur consecutively). All this can be done in time $O(n^2)$.

Note that each time we delete an edge, we can update this information in time $O(n)$: we update the neighbour lists of the edge's endpoints and decrement their

degrees, and if one or both of the endpoints now has degree at most $k+1$ its neighbours are sorted and it is inserted into the correct position in the list of small-degree vertices. From the point of view of updating information, deleting a set of $k+1$ vertices can be regarded as a series of edge-deletions, each performed in time $O(n)$. As there are $O(n)$ edges in total, this means we can perform all updates after deletions in time $O(n^2)$.

Given the array of degrees of all vertices, it is straightforward to identify in time $O(n)$ an edge incident with fewer than $\Delta(G)$ others, if such an edge exists. If there is no such edge, we know from the proof of Theorem 7.7 that (provided the maximum degree of the graph is still at least $(k+1)2^{k+2}$) there exists a set of $k+1$ vertices with a common neighbourhood of size at most $k+1$. As the vertices of degree at most $k+1$ are sorted by their neighbourhoods, it is also possible to identify such a set of vertices in linear time. Thus at each step we are able to identify the edge or set of vertices to delete in time $O(n)$, and so all deletions (and subsequent updating) can be performed in time $O(n^2)$.

It therefore remains to show that we can also perform the reinsertions in time $O(n^2)$. When we reinsert an edge, we can simply colour it with the first available colour from its colour-list, taking time $O(n)$. When reinserting a set of $k+1$ vertices, we need to colour up to $(k+1)^2$ edges which form a complete bipartite subgraph. Recall from the proof of Theorem 7.7 that every such edge still has at least $k+1$ available colours from its list (i.e. colours that have not already been used on incident edges), and that there exists a proper list edge-colouring of the bipartite graph if every edge has a list of exactly $k+1$ permitted colours. For each of the edges, we can compute in time $O(n)$ a list of $k+1$ colours from its colour-list which have not yet been used on incident edges. We can then check in constant time all possible colourings of the $(k+1)^2$ edges in which each receives one of the $k+1$ colours from its list, to find a proper colouring of this bipartite graph, which is guaranteed to extend

the list edge-colouring as required.

Thus we can perform all reinsertions in time $O(n^2)$, completing the proof of the theorem. □

7.2 List Hamilton Path

In the previous section, we saw several examples of problems that are NP-hard in general, yet are solvable in linear time on graphs of bounded treewidth, but which have a W[1]-hard list version (when parameterised by treewidth).

The situation is similar for the problem of determining whether a given graph contains a Hamilton path. Hamilton Path is known to be computationally difficult in general, remaining NP-hard when restricted to planar, cubic, 3-connected graphs (Garey, Johnson and Tarjan [29]) or bipartite graphs (Krishnamoorthy [38]), but can be solved in linear time on graphs of bounded treewidth (Arnborg and Proskurowski [6]). In this section we consider the following list variant of the problem, introduced by Fellows et. al. [23], in which each vertex has a list of permissible positions on the path.

List Hamilton Path
Input: A graph $G = (V, E)$, and a set of lists $\mathcal{L} = \{L_v \subseteq \{1, \ldots, |V|\} : v \in V\}$ of permitted positions.
Question: Does there exist a path $P = P[1] \ldots P[|G|]$ in G such that, for $1 \leq i \leq |G|$, we have $i \in L_{P[i]}$?

Given a graph G and a set of lists $\mathcal{L} = \{L_v \subseteq \{1, \ldots, |G|\} : v \in V(G)\}$ of permitted positions, we say a path $P = P[1] \ldots P[|G|]$ in G is a *valid* Hamilton path if $i \in L_{P[i]}$ for every i.

We show that List Hamilton Path is unlikely to be fixed parameter tractable, even parameterised by pathwidth; this answers another open question of Fellows et. al. [23]. More precisely, we prove the following result.

Theorem 7.10. List Hamilton Path, *parameterised by pathwidth, is W[1]-hard.*

We prove the theorem by means of a reduction from Multicolour Clique (recall the definition of parameterised reductions in Section 1.3). Suppose G is the k-coloured graph in an instance of Multicolour Clique: we may assume without loss of generality that all k vertex classes have the same size, and also that the number of edges between each pair of vertex classes is the same (as adding isolated edges and vertices does not change the existence or otherwise of a multicolour clique). Let the vertex classes be $V_1, \ldots, V_k$, where V_i contains vertices $V_i[1], \ldots, V_i[p]$, and assume that there are q edges between each pair of vertex classes. We describe the construction of a graph H of pathwidth at most $5k$, and a set of lists $\mathcal{L} = (L_v)_{v \in V(H)}$ such that there is a valid Hamilton path in $(H, \mathcal{L})$ if and only if G contains a multicolour clique. Lemmas 7.13, 7.14 and 7.15 will demonstrate that $(H, \mathcal{L})$ does indeed have the required properties.

Our construction consists of $k+1$ paths, with some additional edges linking them: paths $P_1, \ldots, P_k$ correspond to the vertex classes $V_1, \ldots, V_k$, and an additional path Q is used to connect $P_1, \ldots, P_k$. Each path P_i has $2n^2$ vertices (so $|P_i| \geq 4q\binom{k}{2} = 4|e(G))$, and we denote the j^{th} vertex of P_i by $P_i[j-1]$. The path $Q = Q_1 \ldots Q_k$ is the concatenation of k subpaths $Q_1, \ldots, Q_k$, each containing $n^2(n-2)$ vertices and, for $1 \leq i \leq k$, every vertex of Q_i is adjacent to both $P_i[0]$ and $P_i[2n^2 - 1]$.

In addition, we have a number of edge-gadgets, consisting of edges between pairs of the paths $P_1, \ldots, P_k$. Suppose $E(G) = \{e_0, \ldots, e_{m-1}\}$. Then, for each edge e_r between V_i and V_j, we have an edge-gadget $G(e_r)$, involving the r^{th} group of four vertices in P_i and the corresponding group of vertices from P_j. $G(e_r)$ has edges $P_i[4r]P_j[4r+1]$, $P_j[4r]P_i[4r+1]$, $P_i[4r+1]P_j[4r+3]$ and $P_i[4r+2]P_j[4r+2]$, as illustrated in Figure 7.1.

This completes the construction of the graph H. We now define the list of per-

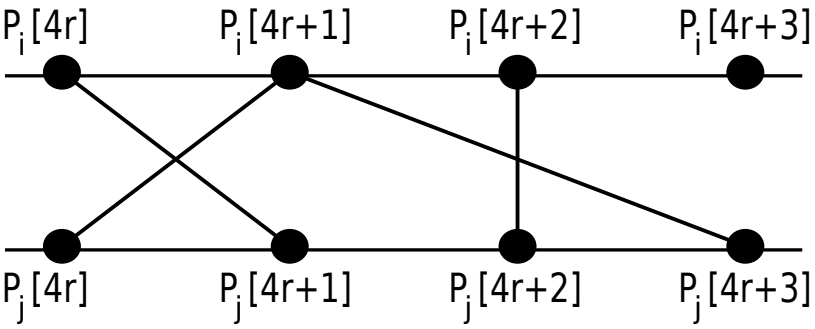


Figure 7.1: The edge-gadget $G(e_r)$

mitted positions for each vertex. For $1 \leq i \leq k$, we set

$$L_{P_i[0]} = \{(i-1)n^3 + 2\alpha n^2 : 1 \leq \alpha \leq p\},$$

and

$$L_{P_i[2n^2-1]} = \{(i-1)n^3 + 2\alpha n^2 + (2n^2-1) + k + 1 - 2i : 1 \leq \alpha \leq p\}.$$

We further define the list

$$L(i,j) = \{(i-1)n^3 + 2\alpha n^2 + j + \beta : 1 \leq \alpha \leq p, -(k-1) \leq \beta \leq k-1\}.$$

For every internal vertex $P_i[j]$, the list $L_{P_i[j]}$ will contain $L(i,j)$. In most cases we in fact set $L_{P_i[j]} = L(i,j)$, the only exceptions being three vertices in each edge-gadget $G(e)$: if $e = V_i[r]V_j[s]$ and the vertices in $G(e)$ are $P_i[l], \ldots, P_i[l+3], P_j[l], \ldots, P_j[l+3]$, then the list for $P_i[l+1]$ will additionally contain positions

$$\{(j-1)n^3 + 2sn^2 + (l+1) + \beta : -(k-1) \leq \beta \leq k-1\},$$

while the lists for $P_j[l+1]$ and $P_j[l+2]$ also contain

$$\{(i-1)n^3 + 2rn^2 + (l+1) + \beta : -(k-1) \leq \beta \leq k-1\}$$

and

$$\{(i-1)n^3 + 2rn^2 + (l+2) + \beta : -(k-1) \leq \beta \leq k-1\}$$

respectively. We place no restriction on the positions that vertices from Q can take in a valid Hamilton path.

The idea is that any valid Hamilton path must (with the exception of a few vertices belonging to the edge-gadgets) traverse $P_1, \ldots, P_k$ in that order, using sections of Q before and after each P_i to connect the paths. In this construction, our choice of position for $P_i[0]$ corresponds to a choice of vertex from V_i: if $P_i[0]$ takes position $(i-1)n^3 + 2rn^2$, we say that it *selects* $V_i[r]$. Note that their lists of permitted positions mean that $P_1[0], \ldots, P_k[0]$ must occur on the path in order, and observe also that every combination of starting positions for these paths can be achieved by linking the paths $P_1, \ldots, P_k$ with vertices from Q. The edge-gadgets allow some vertices to be exchanged between each pair of paths P_i and P_j if the vertices selected by $P_i[0]$ and $P_j[0]$ are adjacent, and we use this, together with conditions imposed by the lists of permitted positions for $P_i[2n^2-1]$ and $P_j[2n^2-1]$, to ensure that the vertices selected by each pair of paths P_i and P_j are adjacent.

We say that a Hamilton path *swaps* in $G(e_r)$ (an edge-gadget containing edges between P_i and P_j) if the segment of the Hamilton path between $P_i[4r]$ and $P_i[4r+3]$ is

$$P_i[4r]P_j[4r+1]P_j[4r+2]P_i[4r+2]P_i[4r+3],$$

and that between $P_j[4r]$ and $P_j[4r+3]$ is

$$P_j[4r]P_i[4r+1]P_j[4r+3],$$

as illustrated in Figure 7.2. Observe that, if $i < j$, swapping in $G(e_r)$ increases the number of vertices lying between $P_i[0]$ and $P_i[2n^2-1]$ on the Hamilton path by one, and correspondingly decreases by one the number of vertices lying between $P_j[0]$ and

$P_j[2n^2-1]$. Note that a partial swap, in which P_i gains vertices from P_j but P_j does not gain any vertex from P_i (or vice versa), is not possible in any Hamilton path, as such a path could not include $P_i[4r+1]$ (or $P_j[4r+1]$ and $P_j[4r+2]$).

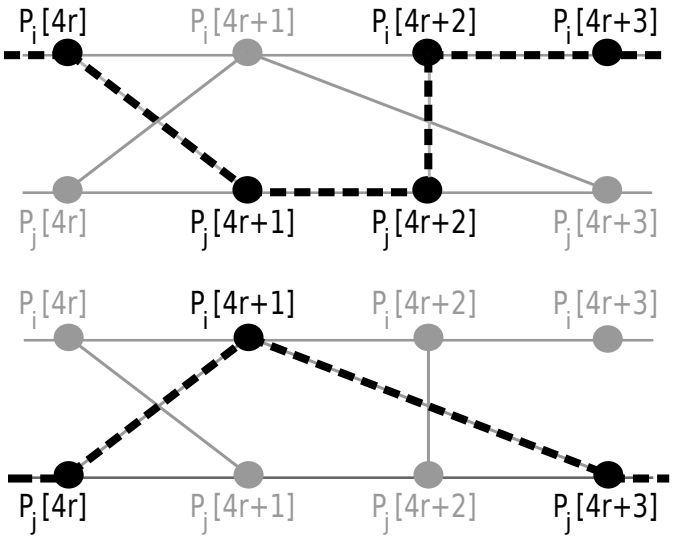


Figure 7.2: A swap in $G(e_r)$

Clearly every vertex of P_i that is not involved in a swap must lie between $P_i[0]$ and $P_i[2n^2-1]$ on any Hamilton path. The range of permitted positions for each vertex $P_i[l]$ (for $1 \leq i \leq k$, $1 \leq l \leq 2n^2-2$) means that, provided P_i is not involved in more than $k-1$ swaps, $P_i[l]$ will appear in a position from its list, no matter where the swaps involving P_i take place. Note also that, if $P_i[l]$ is not involved in a swap, it can only have a position from $\{(i-1)n^3+2rn^2+l+\beta : -(k-1) \leq \beta \leq k-1\}$ if $P_i[0]$ selects the vertex $V_i[r]$.

In the following two propositions, we demonstrate some useful properties of this construction.

Proposition 7.11. *A valid Hamilton path can only swap in the edge-gadget $G(V_i[r]V_j[s])$ if P_i and P_j select vertices $V_i[r]$ and $V_j[s]$ respectively.*

Proof. Suppose that the vertices of $G(V_i[r]V_j[s])$ are $P_i[l], \ldots, P_i[l+3]$ and $P_j[l], \ldots, P_j[l+$

3]. Recall that

$$L_{P_i[l+1]} = L(i, l+1) \cup \{(j-1)n^3 + 2sn^2 + (l+1) + \beta : -(k-1) \leq \beta \leq k-1\}$$

and $L_{P_j[l]} = L(j, l)$. If we swap in $G(V_i[r]V_j[s])$, then $P_i[l+1]$ comes immediately after $P_j[l]$ on the Hamilton path, but if this is not possible if $P_i[l+1]$ takes a position from $L(i, l+1)$, as every element of $L(j, l)$ is strictly larger than every element of $L(i, l+1)$. Thus $P_i[l+1]$ must have a position in $\{(j-1)n^3 + 2sn^2 + (l+1) + \beta : -(k-1) \leq \beta \leq k-1\}$, implying that $P_j[l]$ has a position from the list $\{(j-1)n^3 + 2sn^2 + l + \beta : -(k-1) \leq \beta \leq k-1\}$, which is only possible if $P_j[0]$ selects $V_j[s]$.

Similarly, we see that $P_j[l+1]$ must take a position from $\{(i-1)n^3 + 2rn^2 + (l+1) + \beta : -(k-1) \leq \beta \leq k-1\}$, which implies that P_i selects $V_i[r]$. □

Proposition 7.12. *Any valid Hamilton path in H must involve exactly one swap between each pair of paths from $\{P_1, \ldots, P_k\}$.*

Proof. First note that, by Proposition 7.11, there can be at most one swap between any pair of paths (since at most one edge between two given classes has both its endpoints selected). Now, the list of possible positions for $P_1[2n^2 - 1]$ means we must gain $k-1$ additional vertices on P_1, and this can only be achieved if there are swaps in $k-1$ gadgets involving P_1, implying that there is exactly one swap between P_1 and each other path $P_2, \ldots, P_k$.

Considering P_2, we see that it loses one vertex to P_1, so to achieve its target (a net gain of $k-3$ vertices) it must be involved in at least $k-2$ further swaps. This is the maximum possible, so again implies that in total there is exactly one swap involving P_2 and each other path $P_3, \ldots, P_k$. Continuing in this way we see that we must indeed have exactly one swap between each pair of paths from $\{P_1, \ldots, P_k\}$. □

We now use these facts to demonstrate that $(H, \mathcal{L})$ is a yes-instance for List Hamilton Path if and only if G is a yes-instance for Multicolour Clique.

Lemma 7.13. *If G is a yes-instance for* Multicolour Clique*, then $(H, \mathcal{L})$ is a yes-instance for* List Hamilton Path*.*

Proof. Suppose we have a multicolour clique that contains $V_i[r_i]$ for $1 \leq i \leq k$. We claim we can then find a Hamilton path in H in which every vertex has a position from its permitted list. In such a path, we give $P_i[0]$ position $(i-1)n^3 + 2r_i n^2$, using Q to link up $P_1, \ldots, P_k$ in such a way that each of these paths has the desired starting position. $P_i[0]$ occurs on the path immediately after some vertex $Q_i[\alpha]$ (where, if each path P_j for $j < i$ has a net gain of β_j vertices due to swaps, $\alpha = 2r_i n^2 - 1 - \sum_{j=1}^{i-1} \beta_j$), and $P_i[2n^2 - 1]$ is followed by $Q_i[\alpha + 1]$, so every vertex of Q is included on the path.

We know that, for each pair (i, j) of vertex classes, there exists an edge e_{ij} between $V_i[r_i]$ and $V_j[r_j]$. Thus, for each $1 \leq i < j \leq k$, there exists a gadget $G(e_{ij})$ in which we are able to swap. If we do indeed swap in every such gadget, this gives exactly one swap between each pair of paths, so each path P_i has a net gain of $k + 1 - 2i$ vertices and $P_i[2n^2 - 1]$ has position $(i-1)n^3 + 2r_i n^2 + (2n^2 - 1) + k + 1 - 2i \in L_{P_i[2n^2-1]}$. Each path is involved in exactly $k - 1$ swaps, so all the internal vertices of $P_1, \ldots P_k$ that are not involved in swaps also receive positions from their lists.

This path then includes every vertex in H, and gives each vertex a position from its permitted list, so $(H, \mathcal{L})$ is indeed a yes-instance for List Hamilton Path. □

Lemma 7.14. *If $(H, \mathcal{L})$ is a yes-instance for* List Hamilton Path*, then G is a yes-instance for* Multicolour Clique*.*

Proof. Suppose there exists a valid Hamilton path in H. For $1 \leq i \leq k$, let v_i be the vertex in V_i that is selected by the position of $P_i[0]$ in this path. We claim that the set $\{v_1, \ldots, v_k\}$, which clearly contains one vertex from each colour class, induces a clique in G.

We know, by Proposition 7.12, that in any valid Hamilton path we must have exactly one swap between each pair of paths. But, by Proposition 7.11, a swap can

only occur between P_i and P_j in the gadget $G(uw)$ if u and w are selected by the positions of $P_i[0]$ and $P_j[0]$ respectively. Thus a swap can only occur between paths P_i and P_j if there exists a gadget $G(v_iv_j)$, which – by construction – is only the case if there exists an edge v_iv_j in G. As this holds for every pair (v_i, v_j) with $1 \leq i < j \leq k$, we see that $\{v_1, \ldots, v_k\}$ are pairwise adjacent in G, giving the required multicolour clique. □

Finally, we need to check that H bounded pathwidth.

Lemma 7.15. *H has pathwidth at most $5k$.*

Proof. We construct a path decomposition of H, indexed by a path T with $|T| = |Q| + 2n^2 - 5$. Every bag of the decomposition contains the vertices $V_{end} = \{P_j[0], P_j[2n^2 - 1] : 1 \leq j \leq k\}$. In addition, for $1 \leq i \leq |Q| - 1$, the bag indexed by the i^{th} node of T contains the i^{th} and $(i+1)^{th}$ vertices of Q, while, for $|Q| \leq i \leq |Q| + 2n^2 - 5$, the bag indexed by the i^{th} node of T contains all vertices $\{P_j[i - |Q| + 1], P_j[i - |Q| + 2], P_j[i - |Q| + 3] : 1 \leq j \leq k\}$. Note that every bag contains at most $5k$ vertices.

It is immediate from this construction that, for any vertex $v \in V(H)$, the nodes indexing bags that contain v induce a subpath of T. So it remains to show that, for every edge $uv \in E(H)$, there exists some bag of the decomposition that contains both u and v. For all edges within Q this is clearly true. Note that, for $1 \leq l \leq 2n^2 - 2$, any vertex $P_i[l]$ is only adjacent to vertices $P_j[l']$ where $|l - l'| \leq 2$, and so any edge between internal vertices of the paths $P_1, \ldots, P_k$ must have both its endpoints in some bag of the decomposition. All remaining edges are then incident with some $v \in V_{end}$, but V_{end} is contained in every bag, and so the condition is also satisfied for these edges.

Hence we have a path decomposition of H of width at most $5k$. □

We are now ready to prove the main theorem of this section.

Proof of Theorem 7.10. It follows immediately from Lemmas 7.13 and 7.14 that $(H, \mathcal{L})$ is a yes-instance for List Hamilton Path if and only if G is a yes-instance for Multicolour Clique. H has order polynomial in $|G|$, and can clearly be computed from G in polynomial time. Moreover, by Lemma 7.15 we know that the pathwidth of H depends only on the parameter k, the number of colours used in G. This completes the reduction to show that List Hamilton Path, parameterised by treewidth, is W[1]-hard. □

Chapter 8

Conclusions

The majority of this thesis was concerned with problems related to flood-filling games played on graphs, and contains numerous new results on this subject. Chapters 2 and 3 were dedicated to general results concerning these games, and included a powerful characterisation of the number of moves required to flood a graph G: this is equal to the minimum, taken over all spanning trees T of G, of the number of moves required to flood T. A range of applications of this result were seen in later chapters.

The focus of Chapter 4 was on the relatively rare cases in which flood-filling problems can be solved in polynomial time, and many of the algorithms given here rely heavily on the spanning trees characterisation. We showed that the number of moves required to flood a graph can be computed in polynomial time if the graph has only two colours initially, or if the graph has only a polynomial number of connected subgraphs. We then went on to consider problems in which we want to link only a bounded-size subset of the vertices, and proved that k-Linking-Flood-It is solvable in polynomial time for any fixed k.

Chapter 5 was concerned with the complexity of flood-filling games played on rectangular boards of fixed height, and began with a polynomial-time additive approximation to c-Free-Flood-It on $k \times n$ boards, for any fixed c and k. Next we

improved on a hardness result of Clifford et. al. [16], showing that 4-Fixed-Flood-It and 4-Free-Flood-It are NP-hard when restricted to $3 \times n$ boards (the hardness proof for $3 \times n$ boards in [16] requires an unbounded number of colours). We further demonstrated that Free-Flood-It also remains NP-hard on $2 \times n$ boards, although this construction required an unbounded number of colours. In Chapter 6 we went on to show that c-Free-Flood-It is fixed parameter tractable on $2 \times n$ boards when we take the number of colours c as the parameter.

There remain a number of natural open questions relating to flood-filling games. Firstly, the spanning trees characterisation in Chapter 3 motivates a better understanding of the complexity of flood-filling games on trees. We have seen that Free-Flood-It can be solved in polynomial time on subdivisions on any fixed graph, implying that the problem is polynomially solvable on trees with bounded maximum degree and a bounded number of vertices of degree at least three; it would be interesting to investigate what other classes of trees admit polynomial-time algorithms to solve Free-Flood-It. Additionally, the complexity of 3-Fixed-Flood-It and 3-Free-Flood-It on arbitrary trees remain open. Another open problem concerns the complexity of k-Linking-Flood-It: in Chapter 4 we showed that this problem can be solved in polynomial time for fixed k, but it remains open whether this problem is in fact fixed parameter tractable with parameter k. In the study of flood-filling games played on rectangular boards of fixed height, there remains one key open case, namely the complexity of 3-Fixed-Flood-It and 3-Free-Flood-It when restricted to $3 \times n$ boards. Finally, another direction for further research on flood-filling games would be to consider extremal problems: for example, what colouring ω of a graph G with c colours maximises $m(G, \omega)$? The spanning trees characterisation is likely to be a useful tool for any such investigation.

The remainder of the thesis was concerned with the complexity of list problems on graphs of bounded treewidth. We began by proving structural results which determine

the list edge chromatic number and list total chromatic number of graphs whose maximum degree is sufficiently large compared with their treewidth, thus proving special cases of the List (Edge) Colouring Conjecture and Total Colouring Conjecture. We then used these results to prove that List Edge Chromatic Number and List Total Chromatic Number can be solved in linear time on graphs of bounded treewidth, answering an open question of Fellows et. al. [23]. We also saw that the pattern of NP-hard problems that can be solved in linear time on graphs of bounded treewidth but which have W[1]-hard list versions extends to the problem of finding a Hamilton path in a given graph, showing that List Hamilton Path is W[1]-hard even when parameterised by pathwidth. It would be interesting to investigate whether there are further examples of problems in which this pattern occurs, particularly if such problems also have a natural analogue of list chromatic number type problems.

Bibliography

[1] *Flood it!*, available at https://market.android.com.

[2] *Flood it! 2*, available at http://itunes.apple.com.

[3] *Flood it game*, http://floodit.appspot.com.

[4] *Mad virus*, http://www.bubblebox.com/play/puzzle/539.htm.

[5] Noga Alon, *Restricted colorings of graphs*, Surveys in Combinatorics, 1993, Proceedings, 14th British Combinatorial Conference (K. Walker, ed.), London Mathematical Society Lecture Note Series, vol. 187, Cambridge University Press, 1993, pp. 1–33.

[6] Stefan Arnborg and Andrzej Proskurowski, *Linear time algorithms for NP-hard problems restricted to partial k-trees*, Discrete Applied Mathematics **23** (1989), 11–24.

[7] Sanjeev Arora and Boaz Barak, *Computational complexity: a modern approach*, Cambridge University Press, 2009.

[8] David Arthur, Raphaël Clifford, Markus Jalsenius, Ashley Montanaro, and Benjamin Sach, *The complexity of flood filling games*, Fun with Algorithms, 5th International Conference, FUN 2010, Ischia, Italy, June 2-4, 2010. Proceedings (Paolo Boldi and Luisa Gargano, eds.), Lecture Notes in Computer Science, vol. 6099, Springer, 2010, pp. 307–318.

[9] József Beck, *Combinatorial games: tic-tac-toe theory*, Cambridge University Press, 2008.

[10] M. Behzad, *Graphs and their chromatic numbers*, Ph.D. thesis, Michigan State University, 1965.

[11] Hans L. Bodlaender, *A linear-time algorithm for finding tree-decompositions of small treewidth*, SIAM Journal on Computing **25** (1996), no. 6, 1305–1317.

[12] B. Bollobás and A. Harris, *List-colourings of graphs*, Graphs and Combinatorics **1** (1985), 115–127.

[13] Béla Bollobás, *Modern graph theory*, Springer, 1998.

[14] Richard B. Borie, R. Gary Parker, and Craig A. Tovey, *Automatic generation of linear-time algorithms from predicate calculus descriptions of problems on recursively constructed graph families*, Algorithmica **7** (1992), 555–581.

[15] A. Born, *Flash application for the computer game* Biene (Honey-Bee), http://www.ursulinen.asn-graz.ac.at/Bugs/htm/games/biene.htm, 2009.

[16] Raphaël Clifford, Markus Jalsenius, Ashley Montanaro, and Benjamin Sach, *The complexity of flood filling games*, Theory of Computing Systems (2012), no. 50, 72–92.

[17] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest, *Introduction to algorithms*, MIT Press and McGraw-Hill, 1990.

[18] B. Courcelle, *The monadic second-order logic of graphs I: recognizable sets of finite graphs*, Information and Computation **85** (1990), 12–75.

[19] Gruia Călinescu, Cristina G. Fernandes, and Bruce Reed, *Multicuts in unweighted graphs and digraphs with bounded degree and bounded tree-width*, Journal of Algorithms **48** (2003), 333–359.

[20] R. G. Downey and M. R. Fellows, *Fixed-parameter tractability and completeness I: Basic results*, SIAM Journal on Computing **24** (1995), no. 4, 873–921.

[21] R. G. Downey and M. R. Fellows, *Parameterized complexity*, Springer-Verlag, 1999.

[22] M. Fellows, D. Hermelin, F. Rosamond, and S. Vialette, *On the fixed-parameter intractability and tractability of multiple-interval graph properties*, Theoretical Computer Science **410** (2009), 53–61.

[23] Michael R. Fellows, Fedor V. Fomin, Daniel Lokshtanov, Frances Rosamond, Saket Saurabh, Stefan Szeider, and Carsten Thomassen, *On the complexity of some colorful problems parameterized by treewidth*, Information and Computation **209** (2011), 143–153.

[24] Rudolf Fleischer and Gerard J. Woeginger, *An algorithmic analysis of the honey-bee game*, Theoretical Computer Science **452** (2012), 75–87.

[25] J. Flum and M. Grohe, *Parameterised complexity theory*, Springer, 2006.

[26] Aviezri S. Fraenkel, *Complexity, appeal and challenges of combinatorial games*, Theoretical Computer Science **313** (2004), 393–415.

[27] H. Fukui, A. Nakanishi, R. Uehara, T. Uno, and Y. Uno, *The complexity of free flooding games*, Information Processing Society of Japan (IPSJ) SIG Notes, August 2011, pp. 1–5.

[28] F. Galvin, *The list chromatic index of a bipartite multigraph*, Journal of Combinatorial Theory, Series B **63** (1995), no. 1, 153–158.

[29] M. R. Garey, D. S. Johnson, and R. Endre Tarjan, *The planar hamilton circuit problem is NP-complete*, SIAM Journal on Computing **5** (1976), 704–714.

[30] Michael R. Garey and David S. Johnson, *Computers and intractability: a guide to the theory of np-completeness*, W. H. Freeman, 1979.

[31] Roland Häggkvist and Amanda Chetwynd, *Some upper bounds on the total and list chromatic numbers of multigraphs*, Journal of Graph Theory **16** (1992), no. 5, 503–516.

[32] Ian Holyer, *The NP-completeness of edge-coloring*, SIAM J. on Computing **10** (1981), no. 4, 718–720.

[33] Shuji Isobe, Xiao Zhou, and Takao Nishizeki, *Total colourings of degenerate graphs*, Combinatorica **27** (2007), no. 2, 167–182.

[34] Klaus Jansen and Petra Scheffler, *Generalized coloring for tree-like graphs*, Discrete Applied Mathematics **75** (1997), 135–155.

[35] T. R. Jensen and B. Toft, *Graph coloring problems*, Wiley-Interscience, 1995.

[36] Jeff Kahn, *Asymptotics of the chromatic index for multigraphs*, Journal of Combinatorial Theory, Series B **68** (1996), no. 2, 233–254.

[37] R. M. Karp, *Reducibility among combinatorial problems*, Complexity of Computer Computations (R. E. Miller and J. W. Thatcher, eds.), Plenum Press, New York, 1972.

[38] M. S. Krishnamoorthy, *An NP-hard problem in bipartite graphs*, SIGACT News **7** (1975), 26–26.

[39] A. Lagoutte, M. Noual, and E. Thierry, *Flooding games on graphs*, HAL: hal-00653714, December 2011.

[40] Aurélie Lagoutte, *Jeux d'inondation dans les graphes*, Tech. report, ENS Lyon, HAL: hal-00509488, August 2010.

[41] Daniel Leven and Zvi Galil, *NP completeness of finding the chromatic index of regular graphs*, Journal of Algorithms **4** (1983), no. 1, 35–44.

[42] Dániel Marx, *NP-completeness of list coloring and precoloring extension on the edges of planar graphs*, Journal of Graph Theory **49** (2005), 313–324.

[43] Colin J.H. McDiarmid and Abdón Sánchez-Arroyo, *Total colouring regular bipartite graphs is NP-hard*, Discrete Mathematics **124** (1994), no. 1-3, 155–162.

[44] Kitty Meeks and Alexander Scott, *The complexity of flood-filling games on graphs*, Discrete Applied Mathematics (2012), no. 160, 959–969.

[45] Michael Molloy and Bruce Reed, *A bound on the total chromatic number*, Combinatorica **18** (1998), no. 2, 241–280.

[46] R. Niedermeier, *Invitation to fixed parameter algorithms*, Oxford University Press, 2006.

[47] Panos M. Pardalos and Thelma Mavridou, *The graph coloring problem: A bibliographic survey*, Handbook of Combinatorial Optimization, vol. 2, Kluwer Academic Publishers, 1998.

[48] Kari-Jouko Räihä and Esko Ukkonen, *The shortest common supersequence problem over binary alphabet is NP-complete*, Theoretical Computer Science **16** (1981), 187–198.

[49] D. Seese, *Tree-partite graphs and the complexity of algorithms*, Fundamentals of Computation Theory (Lothar Budach, ed.), Lecture Notes in Computer Science, vol. 199, Springer Berlin / Heidelberg, 1985, pp. 412–421.

[50] V. G. Vizing, *On an estimate of the chromatic class of a p-graph. (Russian)*, Diskret. Analiz **3** (1964), 25–30.

[51] ———, *Some unsovled problems in graph theory (Russian)*, Uspekhi Mat. Nauk **23** (1968), 117–134.

[52] Xiao Zhou, Yuki Matsuo, and Takao Nishizeki, *List total colorings of series-parallel graphs*, Journal of Discrete Algorithms **3** (2005), 47–60.

[53] Xiao Zhou, Shin-ichi Nakano, and Takao Nishizeki, *Edge-colouring partial k-trees*, Journal of Algorithms **21** (1996), 598–617.